



Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

ENSEA

RAPPORT

emotion detector

Élèves :

Mohamed AIT TAKNIOUINE
Yassine EL HAROUCH

Enseignant :

M. Ayoub JEBRI

Table des matières

| | | |
|----------|--|-----------|
| 1 | Abstract | 2 |
| 2 | Introduction | 3 |
| 3 | Partie électronique | 4 |
| 3.1 | Choix des capteurs | 4 |
| 3.2 | La réalisation d'une Printed circuit board (PCB) | 4 |
| 3.3 | Raspberry Pi | 6 |
| 3.4 | Réalisation du montage | 7 |
| 4 | Partie informatique | 8 |
| 4.1 | Voice2text | 8 |
| 4.2 | Modèle | 9 |
| 4.2.1 | Données Utilisées | 9 |
| 4.2.2 | Architecture du Modèle & Entraînement | 12 |
| 4.3 | Résultat et validation du modèle | 13 |
| 5 | Affinement du projet | 16 |
| 6 | Conclusion | 17 |

1 Abstract

Nous proposons un montage constitué d'un microphone (Input), d'une raspberry qui fera l'acquisition et le traitement des entrées via un modèle de traitement de langage naturel (NLP) et d'une PCB (Printed-circuit-board) sur laquelle on ajoutera 4 LEDs de couleurs différentes (on va attribuer à chaque couleur un type d'émotion).

Pour mener à bien ce projet, nous avons tout d'abord listé et analysé les exigences requises, afin d'obtenir une expression précise des besoins. Cette analyse a permis de développer plus efficacement les différentes fonctionnalités. Ensuite, nous avons effectué des tests dans le but de comparer les exigences attendues aux résultats obtenus et d'améliorer ces derniers notamment dans le modèle utilisé. Et enfin, nous avons procédé à un bilan de notre projet.

2 Introduction

Dans le cadre de notre première année du cycle ingénieurs à l'ENSEA, il nous est proposé un projet de 44h 6 semaines(1 mois et demi)nous permettant de mettre en pratique nos connaissances et nos compétences professionnelles au travers d'un cahier des charges ayant pour finalité la détection d'émotion d'une personne.

Ayant une passion commune pour l'intelligence artificielle, notre groupe composé de Mohamed AIT TAKNIOUINE et Yassine EL HAROUCH, a saisi l'opportunité d'exploiter cet intérêt commun pour soumettre l'ébauche d'un projet personnel innovant au responsable M.JEBRI Ayoub.

3 Partie électronique

3.1 Choix des capteurs

Au début on avait l'idée d'utiliser 3 capteurs qui sont les suivants :

- Capteur sonore : microphone afin d'avoir un signal audio qu'on détaillera ensuite son utilité dans la partie informatique pour détecter l'émotion voulue.
- Le capteur de rythme cardiaque (Pulse Sensor Amped) afin d'incorporer une mesure du rythme cardiaque au projet ; ce qui nous permettra de bien savoir l'intensité d'émotion (par exemple : Happy + battement de coeur élevés = very Happy :D)
- L'usage de la détection faciale afin de reconnaître la personne grâce à son visage

Comme on avait que 44h de projet et que notre modèle d'IA était compliqué à réaliser, on a travaillé sur un seul capteur qui est un microphone.

3.2 La réalisation d'une Printed circuit board (PCB)

A l'aide du programme Eagle de CAO PCB pour la conception et la mise en place de conceptions de cartes de circuits imprimés, on a pu réaliser notre PCB qui contient nos Leds. Au début on a commencé par une réalisation d'un montage simplifié afin de savoir les composants dont on aura besoin.

Le schéma est le suivant :

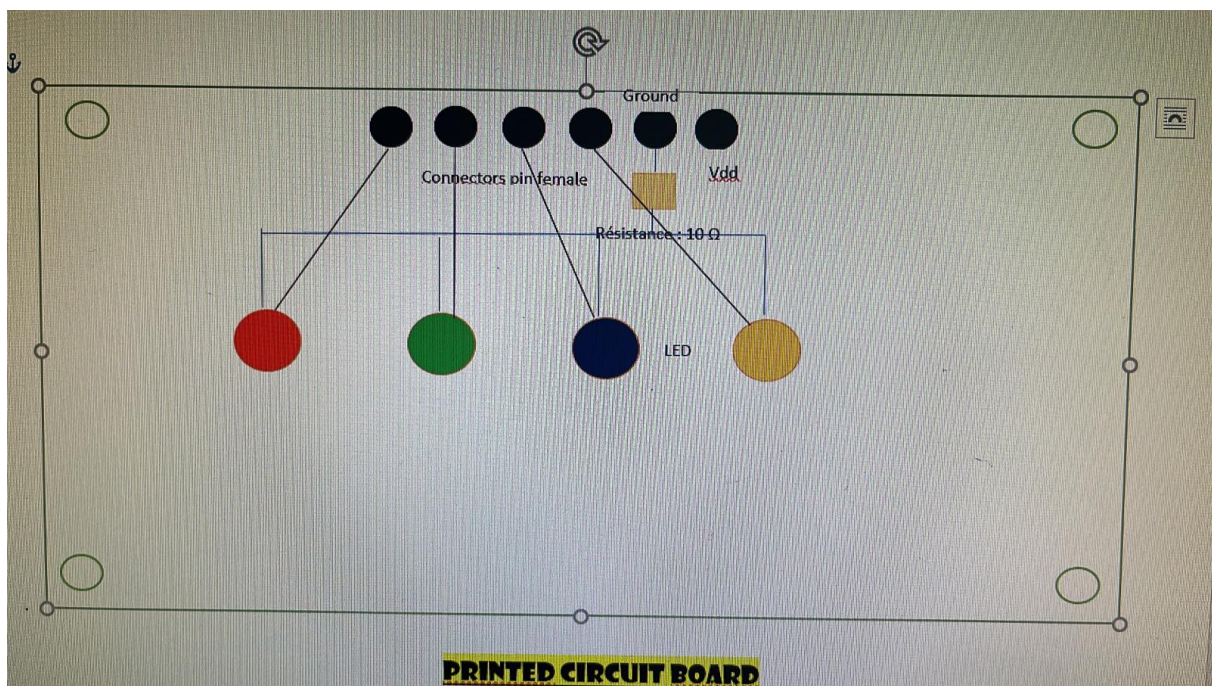


FIGURE 1 – Printed Circuit Board

L'étape suivante était de créer notre schéma sur EAGLE et de faire le routage afin de connecter les composants électriques entre eux.

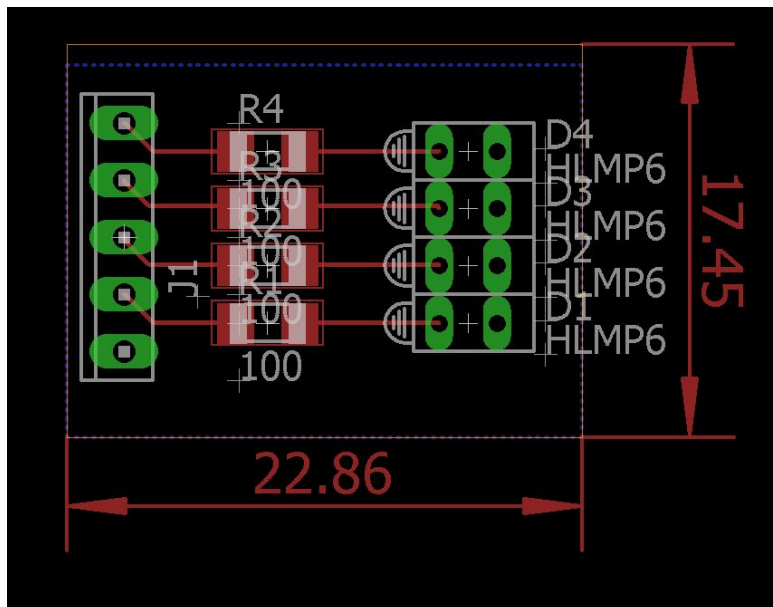


FIGURE 2 – Carte PCB réalisé sur EAGLE

Enfin la construction de notre PCB :

Le technicien dans la salle C-02 était dans l'extrême gentillesse et nous a aidé à imprimer notre PCB.

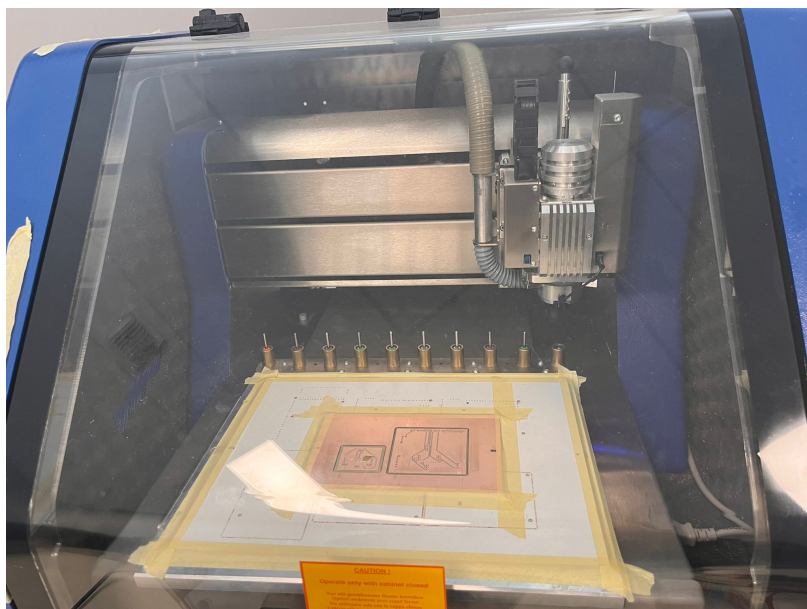


FIGURE 3 – L'impression de la PCB

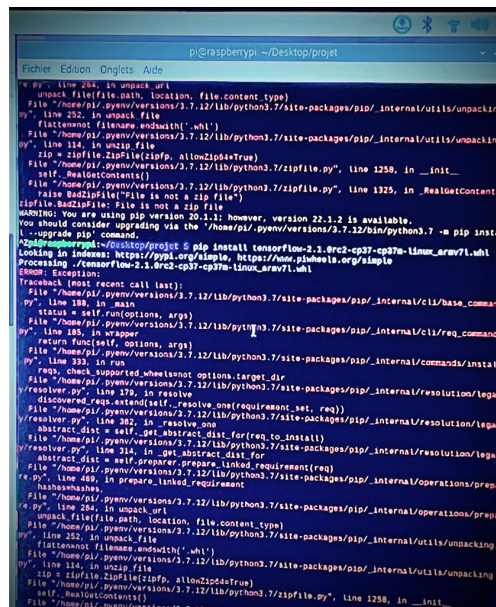
3.3 Raspberry Pi

A l'aide du site : <https://raspberry-lab.fr/Debuter-sur-Raspberry-Francais/Creer-un-programme-Python-Raspberry-Francais/> et d'autres sites du même type, on a pu implémenter notre code python sur notre raspberry ainsi que le faire fonctionner directement en modifiant le fichier `/etc/rc.local`.

Au début et pour faire un petit test on a commencer par un tou petit code pour allumer nos leds successivement, et d'un seul coup ça a marché.

Ensuite on implémente notre modèle et après beaucoup d'essaie ça a marché.

On a rencontré dans cette étape beaucoup de difficultés notamment dans l'installation du tensorflow sur notre raspberry pi 4 comme la montre l'image ci-dessous.



```

pi@raspberrypi: ~/Desktop/projet
Fichier Edition Onglets Aide
* py * line 284, in unpack_url
  unpack_file(file_path, location, file_content_type)
File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/utils/unpacking
py", line 252, in unpack_file
  filename = filename.endswith('.whl')
File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/utils/unpacking
py", line 114, in _unpack_file
  zip = zipfile.ZipFile(zipfp, allowZip64=True)
File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/zipfile.py", line 1258, in __init__
  self._RealGetContents()
File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/zipfile.py", line 1325, in _RealGetContents
  raise BadZipFile("File is not a zip file")
zipfile.BadZipFile: File is not a zip file
WARNING: You are using pip version 20.1.1; however, version 22.1.2 is available.
You should consider upgrading via the '/home/pi/.pyenv/versions/3.7.12/bin/python3.7 -m pip instal
l --upgrade pip' command.
pi@raspberrypi:~/Desktop/projet $ pip install tensorflow-2.1.0rc2-cp37-cp37m-linux_armv7l.whl
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Processing ./tensorflow-2.1.0rc2-cp37-cp37m-linux_armv7l.whl
Exception:
Traceback (most recent call last):
  File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/cli/base_command
.py", line 188, in main
    status = self.run(options, args)
  File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/cli/req_command
.py", line 185, in wrapper
    return func(self, options, args)
  File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/commands/install
.py", line 339, in run
    reqs, check_supported_wheels=not options.target_dir
  File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/resolution/legac
y/resolver.py", line 179, in resolve
    discovered_reqs.extend(self._resolve_one(requirement_set, req))
  File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/resolution/legac
y/resolver.py", line 382, in _resolve_one
    abstract_dist = self._get_abstract_dist_for(req_to_install)
  File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/resolution/legac
y/resolver.py", line 314, in _get_abstract_dist_for
    abstract_dist = self.preparer.prepare_linked_requirement(req)
  File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/operations/prepar
e.py", line 469, in prepare_linked_requirement
    hashes=hashes
  File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/operations/prepar
e.py", line 254, in unpack_url
    unpack_file(file_path, location, file_content_type)
File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/utils/unpacking
.py", line 252, in unpack_file
  filename = filename.endswith('.whl')
File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/site-packages/pip/_internal/utils/unpacking
.py", line 114, in _unpack_file
  zip = zipfile.ZipFile(zipfp, allowZip64=True)
File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/zipfile.py", line 1258, in __init__
  self._RealGetContents()
File "/home/pi/.pyenv/versions/3.7.12/lib/python3.7/zipfile.py", line 1325, in _RealGetContents
  raise BadZipFile("File is not a zip file")
zipfile.BadZipFile: File is not a zip file

```

FIGURE 4 – Erreur dans l'installation du tensorflow

3.4 Réalisation du montage

Pour réaliser le montage, on a commencé en premier temps par réaliser un montage équivalent pour tester si ça marche à l'aide d'une plaque d'essai comme montre la figure ci-dessous :

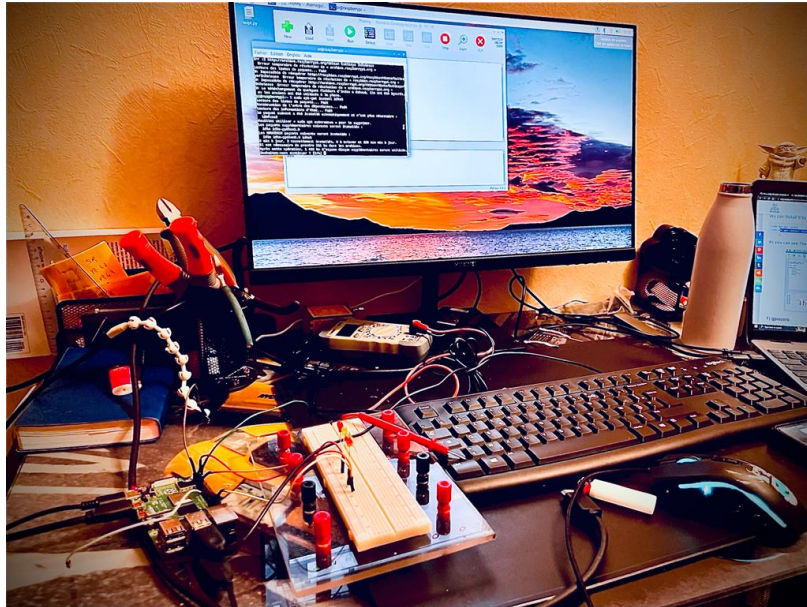


FIGURE 5 – Montage de test

Ensuite on a passé au circuit "réel" qu'a réalisé à l'aide de notre PCB en passant par plusieurs étapes tels que soudage des Leds liaison des composants etc.

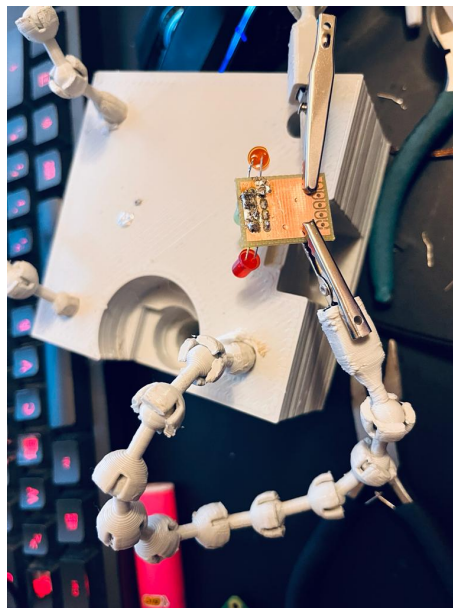


FIGURE 6 – Réalisation du montage

4 Partie informatique

4.1 Voice2text

Notre Input était un microphone qui a comme but de convertir les vibrations sonores dans l'air en signaux électroniques. Ces derniers seront des signaux qu'on veut convertir en texte (des paroles).

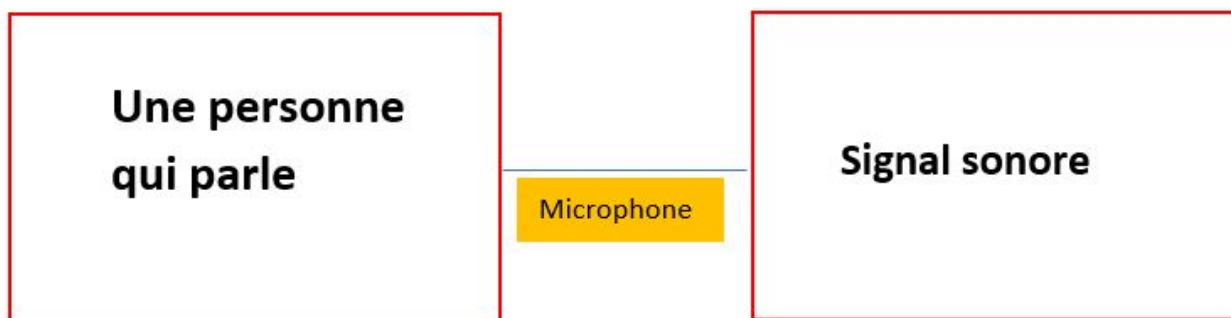


FIGURE 7

A l'aide du code python suivant on a pu donc avoir comme sorti un texte quand analysera ensuite via notre modèle afin de conclure le type d'émotion.

```
22.8s
!pip install nltk
!pip install SpeechRecognition
!pip install deep-translator

{...}

import speech_recognition as sr
import nltk
from nltk.stem import WordNetLemmatizer
from deep_translator import GoogleTranslator

Vocal_happiness = "Vocal happiness.wav"
Vocal_anger = "Vocal anger.wav"
Vocal_fear = "Vocal fear.wav"
Vocal_sadness = "Vocal sadness.wav"

r = sr.Recognizer()
nltk.download('wordnet')
nltk.download('omw-1.4')
```

FIGURE 8 – Code

```
def decodeSpeech(wavefile):
    r = sr.Recognizer()
    with sr.WavFile(wavefile) as source:
        audio = r.record(source)
        try:
            print('Transcription GOOGLE: ' + r.recognize_google(
                audio, language='fr-FR', show_all=False))
        except LookupError:
            print('Cannot understand audio!')

        try:
            print('Transcription SPHINX: ' + r.recognize_sphinx(
                audio, language='fr-FR', show_all=False))
        except sr.UnknownValueError:
            print('Sphinx could not understand audio')
        except sr.RequestError as e:
            print('Sphinx error: {}'.format(str(e)))

decodeSpeech(Vocal_happiness)
decodeSpeech(Vocal_anger)
decodeSpeech(Vocal_fear)
decodeSpeech(Vocal_sadness)
```

```
Traceback (most recent call last):
  at block 5, line 19
NameError: name 'Vocal_happiness' is not defined
import Vocal_happiness
```

```
with open('happiness.txt') as f:
    contents = f.read()
    lemmatizer = WordNetLemmatizer()
    happiness = contents.split('\n')
    hapiness = [lemmatizer.lemmatize(word) for word in happiness]
    print(lemmatizer.lemmatize(text), happiness)
```

```
Traceback (most recent call last):
  at block 6, line 6
NameError: name 'text' is not defined
import text
```

```
to_translate = 'I want to translate this text'
translated = GoogleTranslator(source='en', target='fr').translate(to_translate)
print(translated)
```

Je veux traduire ce texte

FIGURE 9 – Suite du code

4.2 Modèle

4.2.1 Données Utilisées

On a importé des données à l'aide de la fonction load dataset de la bibliothèque nlp.

```
dataset = nlp.load_dataset('emotion')
```

Notre dataset émotion à la forme suivante :

```
{'train': Dataset(features: {'text': Value(dtype='string', id=None), 'label': Value(dtype='string', id=None)}, num_rows: 16000),
 'validation': Dataset(features: {'text': Value(dtype='string', id=None), 'label': Value(dtype='string', id=None)}, num_rows: 2000),
 'test': Dataset(features: {'text': Value(dtype='string', id=None), 'label': Value(dtype='string', id=None)}, num_rows: 2000)}
```

FIGURE 10

Puis on divisé les données à :

- Donnée par Entraînement ; `train = dataset['train']`
- Donnée par Validation ; `val = dataset['validation']`
- Donnée par Test ; `test = dataset['test']`

Le but ensuite était de prédire les labels à partir des tweets ; `tweets, labels = get_tweets(train)`

```
a = tweets[3]
b = labels[3]
a,b #exemple le tweet a est lié à l'emotion b

('i am feeling grouchy', 'anger')
```

FIGURE 11

On passe après à la partie du traitement de données qui a pour but de transformer les données tweets en mots part mots à l'aide de la fonction tokenizer comme montre le code ci dessous :

```
from tensorflow.keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer(num_words=10000, oov_token='<UNK>')
tokenizer.fit_on_texts(tweets)
print(tokenizer.texts_to_sequences([tweets[10]]))

[[2, 3, 310, 446, 2, 95, 31, 3799]]
```

FIGURE 12

Visualisation des données sous forme d'histogramme :

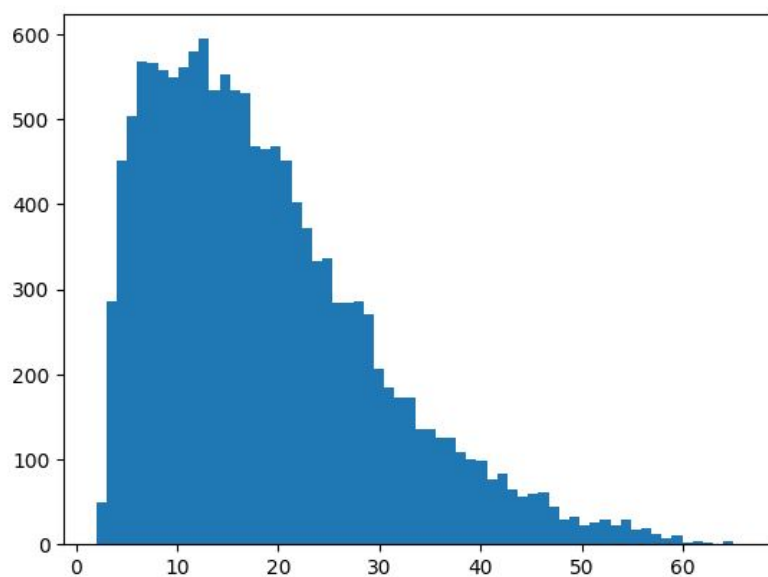


FIGURE 13

On définit ensuite les labels(émotions à détecter) : 'joy', 'anger', 'sadness', 'fear'

On visualise les fréquences des données pour chaque label :

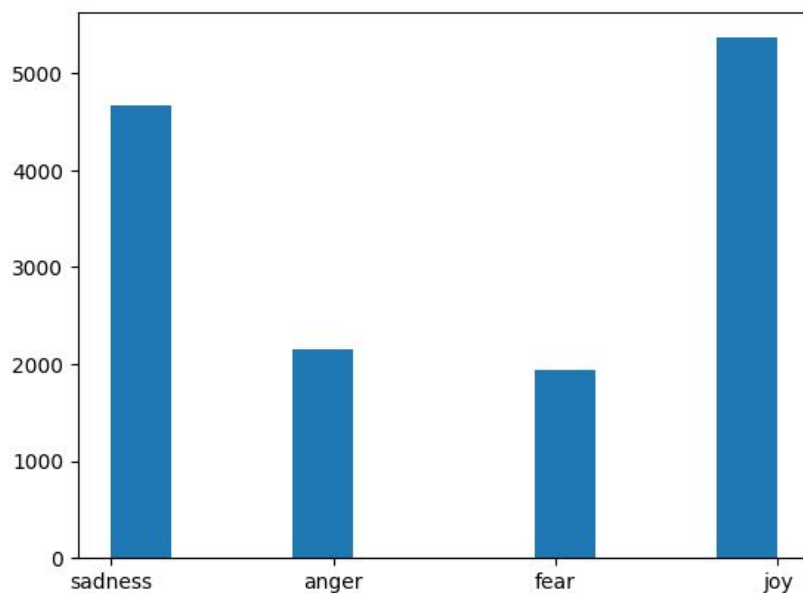


FIGURE 14

4.2.2 Architecture du Modèle & Entraînement

On crée notre modèle qui sera un réseau de neurones (10000 neurones - LSTM (20 neurones) - LSTM (20 neurones) - Couche finale avec 6 neurones auxquels on applique la fonction d'activation softmax)

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Embedding(10000, 16, input_length=50),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(20, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(20)),
    tf.keras.layers.Dense(6, activation='softmax')
])
```

FIGURE 15

Pour la méthode d'entraînement, on utilise une rétropropagation optimisée càd adam optimizer pour l'entraînement et la détermination des paramètres du modèle.

```
model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
model.summary()
```

FIGURE 16

On entraîne finalement le modèle en utilisant la training set constituée de : entrées : padded train sequences | variable à prédire : train labels

```
= model.fit(
    padded_train_sequences, train_labels,
    validation_data=(val_sequences, val_labels),
    epochs=15,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=2)
    ]
)
```

FIGURE 17

4.3 Résultat et validation du modèle

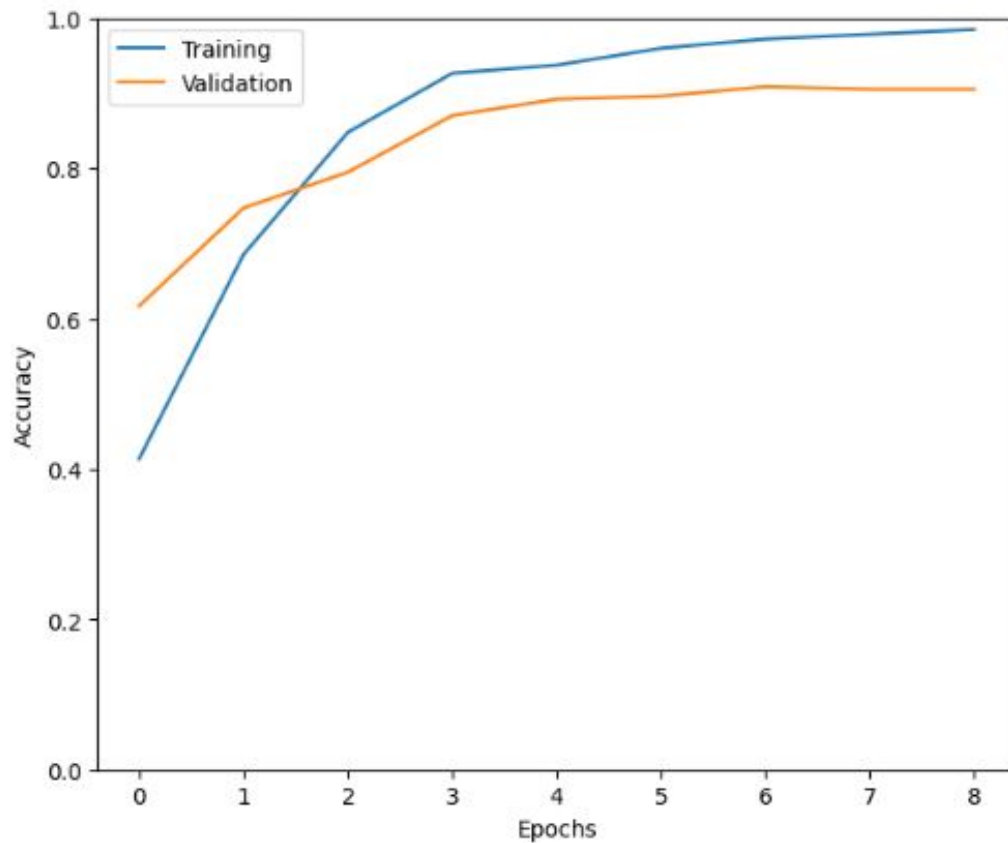


FIGURE 18

Epochs est un paramètre qui gère l'entraînement du modèle.

- Dans notre cas, on a un epochs modéré donc un modèle efficace et bien entraîné sans overfitting ni underfitting.

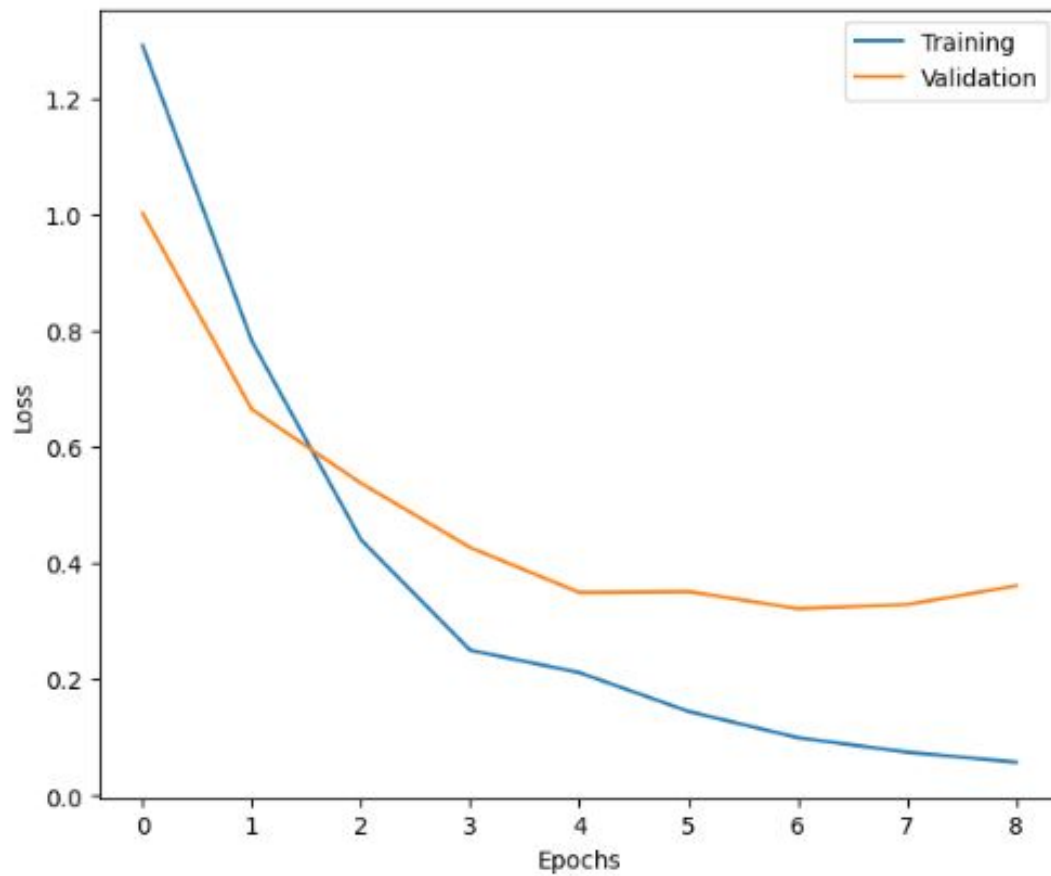


FIGURE 19

- On a un epochs grand donc une perte d'information minime donc modèle efficace avec une compréhension suffisante des détails sans plus, ni moins.

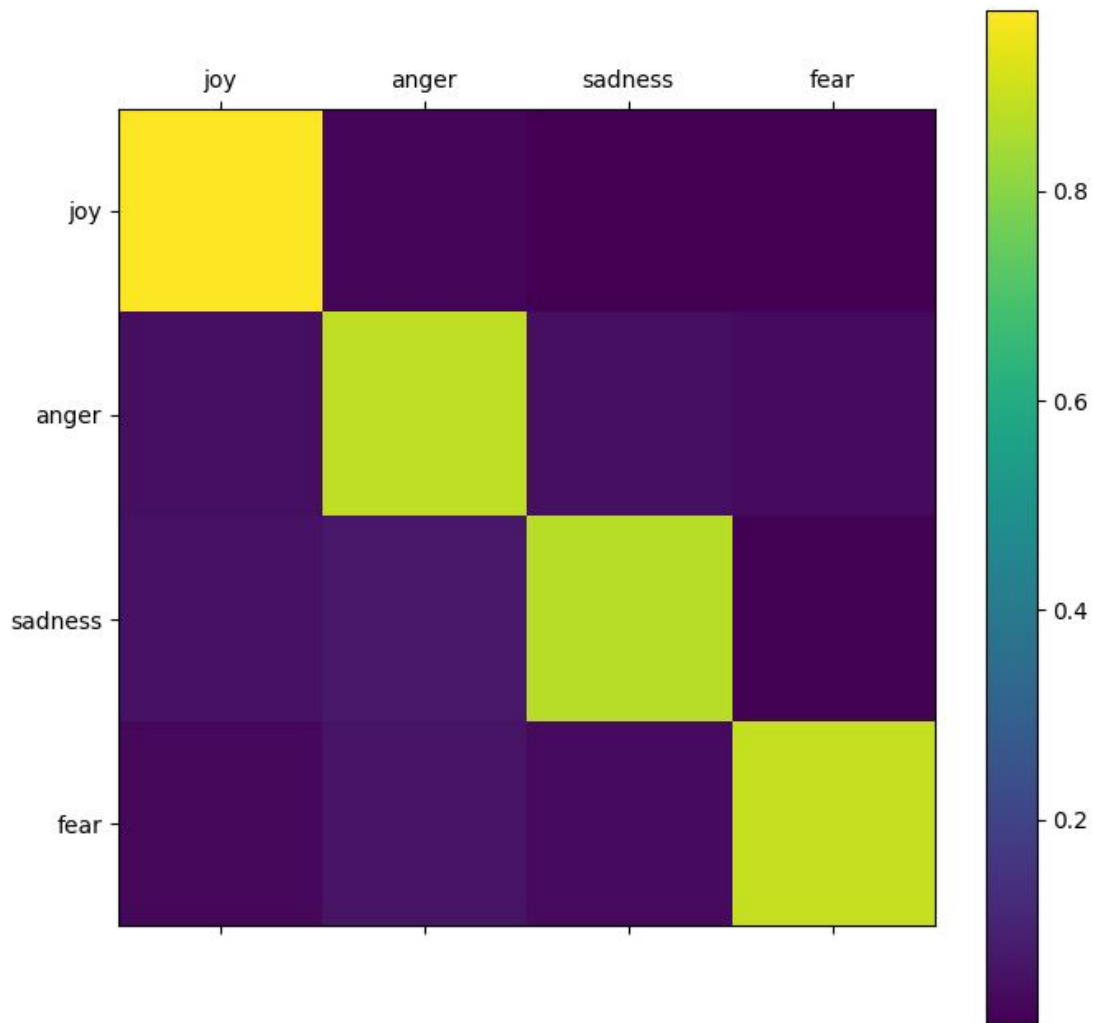


FIGURE 20

La matrice de confusion permet de montrer que la corrélation entre les labels prédits et les labels réels dans l'ensemble de validation coïncident et ceci s'explique par le fait que la corrélation entre deux labels identiques (labels réel-label prédit) est assez proche de 1, ce qui montre que le modèle est efficace en terme de prédiction, c'est ce qu'on veut !

5 Affinement du projet

Comme la détection des émotions peut se faire via plusieurs capteurs, cela va nous permettre d'augmenter les chances que le résultat obtenu soit effectivement celui qu'on attend.

On peut faire ça on ajoutant par exemple :

- La reconnaissance d'états émotionnels par analyse visuelle du visage (les six émotions d'Ekman). Les données correspondantes consistent en des images de visages en niveaux de gris de 48x48 pixel.
- Un capteur d'émotions en mesurant à distance notre rythme cardiaque et notre pouls ; Dans ce contexte des chercheurs du MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) ont développé un capteur sans fil en mesure de détecter les émotions des personnes à distance, en analysant les changements dans leur rythme cardiaque et leur respiration. Je vous invite à jeter un oeil sur leur article qui paraît très intéressant :
https://news.mit.edu/2016/detecting-emotions-with-wireless-signals-0920* .

On peut aussi améliorer notre projet en ayant comme sortie (Output) un message qui s'affiche sur un écran et qui montre notre type d'émotions ainsi que son intensité.

Une idée : c'est de faire entrainer notre modèle sur une évolution d'émotions. Cela va t-il nous permettre de deviner l'émotion suivante ?

6 Conclusion

Ce projet nous a permis de savoir en premier lieu, bien répartir les tâches en fonction du temps, en deuxième lieu de faire une PCB et de nous familiariser et prendre la main avec le logiciel Eagle.

Je trouve que le grand avantage de ce projet, c'est que nous avons appris plein de techniques et méthodes orientée Data et notamment la mise en place d'un modèle de traitement du langage naturel ainsi que son entraînement et l'évaluation de son efficacité.

On a essayé de rédiger ce rapport en \LaTeX pour se familiariser plus avec la méthodologie de la production de papiers de recherche, ce qui nous aidera plus tard.