BORDEAUX INP Ensc

MusiClassify

Rapport final - MusiClassify

Gonçalves Tristan

ENSC - École Nationale Supérieure de Cognitique

E-mail:

- tristan.goncalves@ensc.fr

21 Avril 2023

I. Contextualisation

Ayant toujours aimé écouter de la musique, il y a peu de styles que je n'apprécie pas. En revanche, j'ai souvent du mal à distinguer certains genres, qui peuvent se ressembler mutuellement.

C'est pourquoi l'idée de créer un algorithme de machine learning m'est venue à l'esprit : si je n'arrive pas à faire ces distinctions, peut-être qu'un programme informatique y arrivera mieux que moi ?

Par conséquent, l'objectif de ce projet informatique individuel était de créer un modèle de machine learning permettant d'identifier le genre/style de musique d'un fichier audio qui lui serait fourni, sur la base des 10 styles différents présents dans le jeu de données initial.

II. Choix techniques

A. Avant-propos

Par souci de confort de lecture, les prochaines pages présenteront de façon succincte certaines des fonctions codées. Afin de plonger plus en détail dans ces dernières, le code est fourni avec le rapport. Le code “brut” est présent, ainsi qu’un notebook, permettant une navigation et une utilisation plus aisées et “accompagnée”. Ce dernier est facilement accessible sur **Kaggle**.

De plus, le code source étant trop lourd pour le dépôt sur Moodle ou l’envoi par mail, les différents scripts peuvent être trouvés sur le repo GitHub suivant:

MusiClassify
ou sur ce **lien Drive**
ou sur ce dépôt **RENATER**

Attention : le code sur le repository ne contient pas tout le dataset, il ne fonctionnera pas correctement. Pour utiliser le code, privilégiez l’archive ou le notebook.

Dans l’éventualité où le code serait utilisé sans notebook, il faut d’abord lancer la commande

```
pip install -r requirements.txt
```

afin d’installer tous les packages nécessaires.

Ensuite, le fichier `main.py` permet d’effectuer les différentes étapes pour le traitement des données ainsi que la création et l’entraînement du modèle.

Enfin, afin de tester certains de mes modèles sur quelques musiques “réelles”, il faut lancer le fichier `test.py` situé dans le répertoire **test_real_music/**.

B. Choix du jeu de données (dataset)

La première étape dans un projet d'apprentissage est la sélection ou récolte des données pouvant être utilisées.
Pour ce qui est de la classification de genres musicaux, un dataset connu existe: le *GTZAN*. Ce dernier est composé de 1.000 enregistrements audios de 30 secondes, répartis en 10 genres différents : *blues, classique, country, disco, hiphop, jazz, métal, pop, reggae* et *rock*.
Les enregistrements audios sont en format **wav**, connu pour porter plus d'informations que les formats mp3, grâce à l'absence de compression du signal audio.

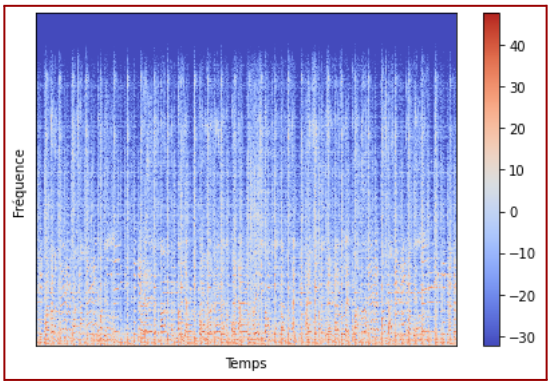
C. Pré-traitement des données

Le dataset GTZAN, bien que connu dans le traitement de musiques, ne contient pas énormément de données pour l'entraînement d'un modèle. De ce fait, il a fallu pré-traiter les données en faisant de la *data augmentation* (augmentation de données).
Cependant avant cela, il a fallu vérifier que toutes les données soient utilisables. Il s'est avéré qu'un des morceaux de la catégorie `jazz` était corrompu. Inexploitable, j'ai été obligé de le supprimer.
Ensuite, les morceaux audio ont été rognés en plus petits extraits. Plusieurs tentatives ont été effectuées pour déterminer une bonne longueur de découpe. D'abord, des extraits de 3 secondes (10 divisions par morceau, multiplication des données par 10) ont été créés, mais les résultats finaux ne paraissant pas assez bons, le nombre de découpes a été descendu à 7, puis 6, afin d'obtenir des extraits de 5 secondes (pour que les extraits aient tous une durée identique).

D. Traitement des extraits

Une fois les audios découpés comme je le souhaitais, une nouvelle question s'est posée : "*Comment traite-t-on des données audio ?*"
La façon la plus évidente serait d'utiliser les spectrogrammes, mais une autre option est apparue après recherches.

1. Spectrogrammes



Pour un sujet de traitement de musiques, il est évident que nous nécessitons des informations de fréquences, mais également des informations concernant le temps.

Dans le monde du traitement de signal, l'objet répondant à ces deux critères est le spectrogramme. En effet, il permet d'apporter des informations concernant temps et fréquences sur un audio, contrairement à une simple transformée de Fourier qui n'indique aucune information relative au temps.

2. MFCCs

De façon plus spécifique au projet de classification de genres musicaux, on peut utiliser les MFCCs, ou coefficients cepstraux de fréquences Mel.

Le calcul de ces derniers utilise la mise à l'échelle Mel ("Mel" pour mélodie), qui représente mathématiquement une approximation de la perception humaine d'un signal audio. En effet, l'être humain ne perçoit pas les sons de façon linéaire mais plutôt sous une forme logarithmique non linéaire.

J'ai donc extrait les MFCCs de mes différents morceaux et les ai stockés dans un fichier *json* ayant pour forme :

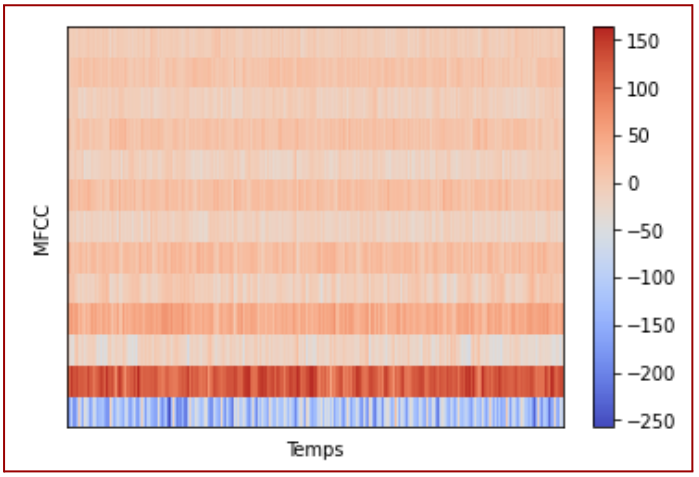
```
data_mfcc = {
   "mapping": [
    "blues", "classical","country","disco","hiphop","jazz","metal","pop","reggae","rock"
   ],
   "mfcc": [...],
   "labels": [...]
}
```

Ici, "mapping" représente les différents genres musicaux disponibles, "mfcc" contient les MFCCs des différents extraits, et "labels" contient des chiffres de 0 à 9, associés à chaque extrait et représentant les différents genres retrouvés dans "mapping".

Pour effectuer cela, j'ai codé une fonction permettant d'extraire les mfcc d'un fichier en particulier (utile pour la suite lors de la prédiction), et une autre utilisant cette dernière afin de traiter toutes les données du dataset.

file_mfcc	Extrait les MFCC d'un fichier fourni et les écrit dans un json donné.
save_mfcc	Utilise ***file_mfcc*** pour traiter le jeu de données entier.

Une représentation graphique des MFCC est de la forme :



III. Architecture du réseau

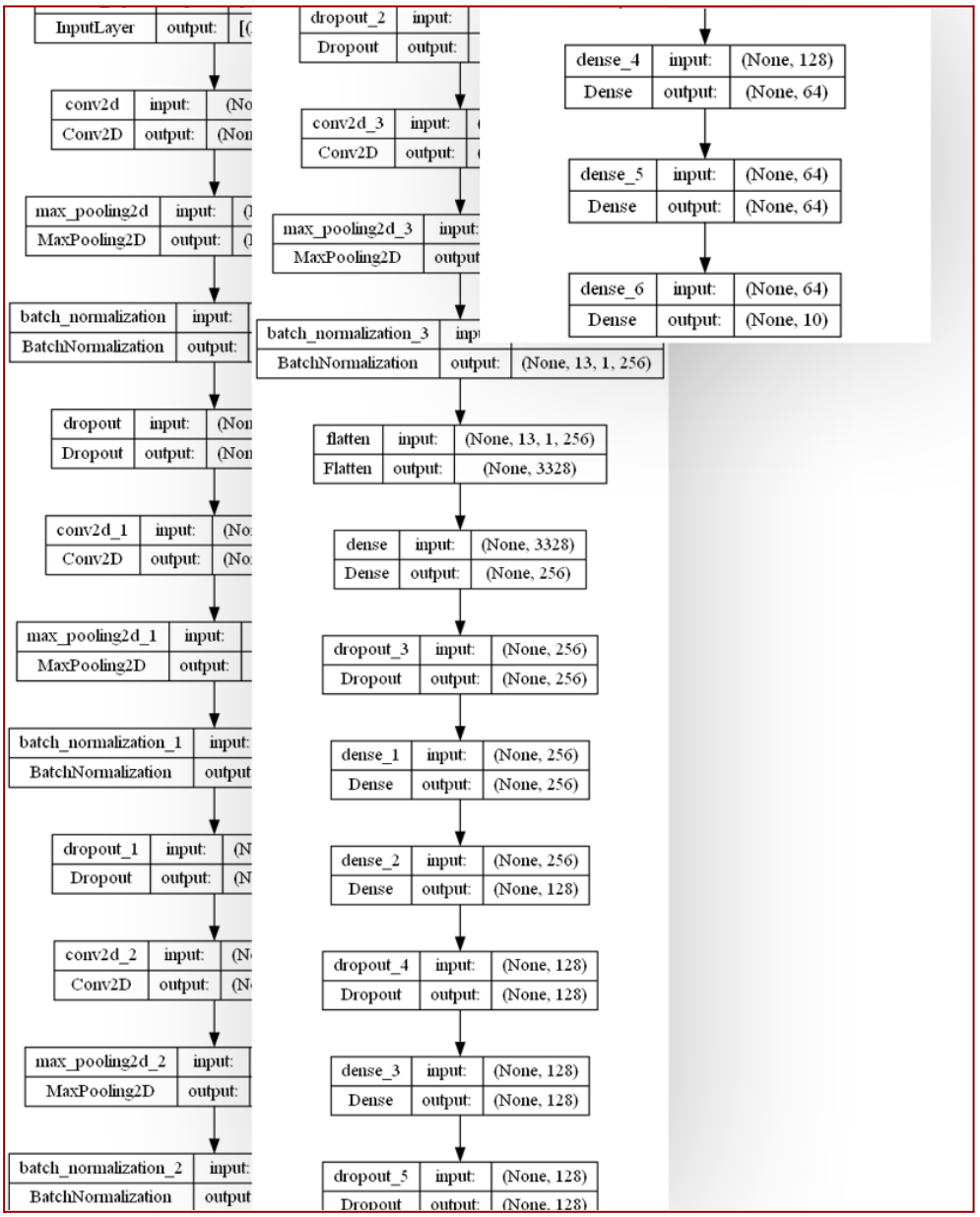
A. Choix du type de réseau

Les MFCC n'étant que des matrices de nombres, cela peut se traiter comme une image. Pour cela, il est commun d'utiliser un réseau neuronal convolutif (CNN).
En effet, les CNN sont des réseaux neuronaux habituellement utilisés pour de la reconnaissance d'images et de sons. Ils sont composés de plusieurs types de couches, notamment les couches de convolution et de pooling. Ces deux couches vont être alternées, la couche de convolution extrayant des caractéristiques, et la couche de pooling réduisant ces cartes de caractéristiques en y appliquant des transformations mathématiques.

B. Modèle construit

Il y a eu plusieurs tentatives avant de parvenir au modèle actuel. En effet, les modèles précédents montraient des résultats très faibles, avec des précisions entre 60 et 65%. Le modèle actuel semble un peu plus performant, avec une précision supposée de 76.67%.

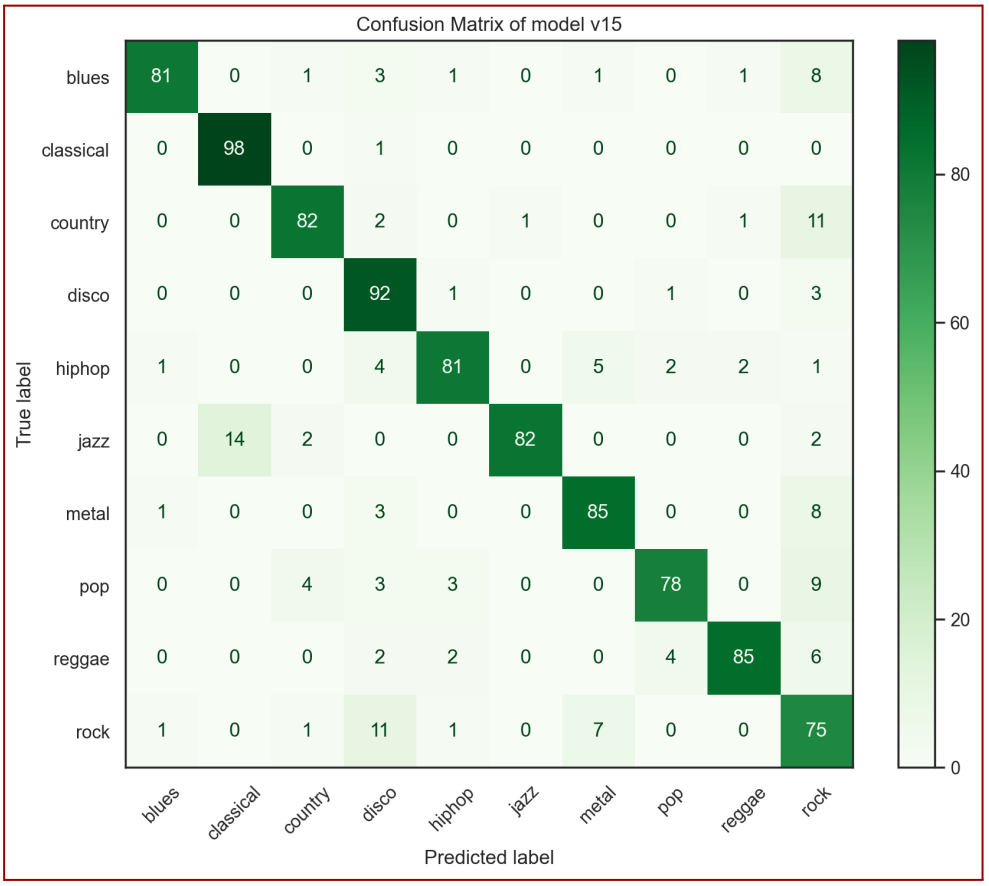
Ci-dessous est un schéma représentatif du modèle et des différentes couches qui le composent.



Sur ce schéma sont visibles les alternances évoquées précédemment entre couches de convolution et couches de pooling. De plus, on aperçoit des couches “Dropout” qui ont pour but de supprimer un certain taux de neurones lors des entraînements, permettant de rendre le modèle plus robuste en lui faisant varier les neurones qu’il fait travailler. Après les différentes couches de convolution sont présentes les couches

“Dense”, qui sont là pour traiter les données issues des précédentes couches et enfin classifier selon les genres recherchés.

Voici la matrice de confusion du dernier modèle en date.



En colonnes sont représentés les genres prédits par le modèle, en opposition avec, en ordonnée, les labels réels des morceaux traités. Cette matrice représente donc, dans sa diagonale, le pourcentage de bonnes prédictions du modèle sur le set de `test` (sur lequel il ne s’est pas entraîné).

Ce genre de représentation sert de validation du modèle auprès des données. En effet, il semblerait que le modèle soit assez précis sur les échantillons du set de test.

C. Autres méthodes/fonctions diverses

Un ensemble d’autres méthodes a été créé, servant à différentes échelles dans le projet.

	Fonction	Description
Traitement des audio	**audio_to_wav**()	Permet de convertir un fichier audio sous format mp3 à un fichier sous format wav.
	mp3_to_wav_all_files()	Convertit tous les fichiers audio présents dans un dossier en wav.
	trim_audio()	Coupe un audio selon un temps de début et un temps de fin fournis par l'utilisateur.
Préparation des données	**prepare_dataset**()	Utilise le *train_test_split* de scikit-learn afin de séparer le dataset en trois sets différents: - le set d'entraînement, - le set de validation, servant durant l'entraînement du modèle, - le set de test, permettant d'avoir un set qui n'est réellement jamais vu par le modèle durant son entraînement.
	load_data()	Comme son nom l'indique, permet de charger les données contenues dans le fichier *json* contenant les MFCCs.
	split_data_file()	Permet la découpe du dataset (format json) en tableaux numpy pour les données d'entraînement, validation et test. Les sauvegarde en fichiers ensuite.
Création modèle	**build_model**()	Va tout simplement créer notre modèle avec un réseau de convolution.
Visualisation	**plot_loss_acc**()	Trace les courbes de précision et de perte du modèle après son entraînement.
	plot_conf_mat()	Affiche la matrice de confusion du modèle fourni en entrée.
Prédiction	**predict**()	Affiche la prédiction du modèle sur des MFCCs fournis.
	test_range()	Permet d'effectuer une série de tests afin de jauger de la prédiction sur un nombre d'essais défini par l'utilisateur.
Autre	**most_common**	Fonction `lambda` (une seule ligne) permettant de récupérer l'élément avec le plus d'occurrence dans

		une liste.
Exclusives au notebook	**conf_matrix_on_chosen_version**()	Affiche la matrice de confusion de la version choisie par l'utilisateur dans le menu déroulant.
	process()	Effectue la prédiction sur la musique choisie par l'utilisateur.

D. Analyse d'une musique "réelle"

Afin de tester le modèle sur des chansons "réelles", j'ai utilisé différentes fonctions pour pouvoir faire une classification la plus complète possible.

En effet, en écoutant de la musique de façon générale, on peut se rendre compte que dans beaucoup de cas, les secondes du départ servent principalement à la "mise en situation" de l'utilisateur. Ces secondes n'ont pas systématiquement une signature de style bien représentative de la musique globale. C'est pourquoi j'ai fait le choix de n'analyser les morceaux qu'après les 30 premières secondes. Ceci permet de limiter les erreurs d'évaluation que pourraient engendrer ces départs pas toujours pertinents.

Ensuite, une fois ces 30 secondes dépassées, on va analyser la chanson en la coupant en autant de morceaux de 30 secondes que possible. Par exemple, une chanson de 2 minutes sera traitée comme tel :

- 30 premières secondes ignorées
- Découpage de 3 extraits de 30 secondes sur la minute et demie restante.

Enfin, chacun de ces morceaux sera traité de la même manière que citée dans le rapport, à savoir la recherche des MFCCs, puis le passage dans le modèle avec la fonction predict.

IV. Support : plateforme web

A. MusiClassifyWeb

MusiClassifyWeb, trouvable sur ce repository, est la plateforme web permettant d'utiliser le modèle de façon plus aisée et graphiquement plus agréable.

Afin d'utiliser le modèle sans encombre, j'ai utilisé le framework Flask pour développer ce site. Flask est un framework léger permettant de développer un site internet en

utilisant python. De ce fait, les fonctions codées durant le projet informatique sont directement implémentables dans le site.

Le but premier était de déployer la plateforme web afin que n'importe qui puisse y accéder sur internet.
Cependant, certains problèmes se sont présentés à moi (cf. Problèmes rencontrés), il faut donc le télécharger (ou le cloner) depuis le repository, afin de l'utiliser en localhost.

Pour lancer le site une fois téléchargé, il faut ouvrir l'invite de commande et se placer dans le dossier contenant les fichiers. Ensuite il faut créer son environnement virtuel ou se connecter à celui fourni

```
Créer        python - m venv {nom_de_l_environnement}

Connecter            venvWindows/Scripts/activate
```

On peut ensuite installer les packages nécessaires avec

```
pip install -r requirements.txt
npm install
```

Enfin,

```
flask --app myApp run
```

Il vous sera demandé de choisir une version de modèle entre la version 6 et la 15. Une fois le choix fait, le site va démarrer et sera accessible à l'adresse

http://127.0.0.1:5000

V. Problèmes rencontrés

Premièrement, le plus gros facteur de ralentissement du projet a été la reprise en main de python, ainsi que l'apprentissage et la compréhension en simultané des méthodes de machine learning ainsi que celles de traitement audio.

Ensuite, il peut être intéressant de noter que j'ai créé une fonction que *je n'ai pas le droit d'utiliser*. En effet, il semblait intéressant de pouvoir créer un "*scraper*" afin d'aller télécharger de façon automatique des fichiers audio sur internet pour des tests de modèle ou enrichir ce dernier. Je n'ai rien vu sur les CGU du site visé indiquant l'interdiction de l'utilisation de tels codes.
Après avoir codé et testé une première fois, malgré le fait que le site fournisse ces chansons de façon gratuite, la question de l'autorisation d'utilisation m'est revenue à l'esprit. De ce fait, j'ai envoyé un message au support afin d'en savoir plus et de me rassurer. Malheureusement, le support m'a indiqué que dû à des utilisations de *scraper* de façon commerciale, il était désormais interdit d'utiliser de tels moyens sur leur site. J'ai donc dû abandonner l'utilisation de ce programme.
Ils m'ont néanmoins souligné qu'ils m'en diraient plus à la fin de l'année, je leur demanderai donc à nouveau à ce moment-là, avec pour objectif de continuer ce projet.

Le dernier point négatif à soulever est le fait que je n'ai pas pu déployer mon modèle. En effet, celui-ci est trop lourd pour les services d'hébergement gratuits, tels que PythonAnywhere ou Vercel. De ce fait, le site Flask est utilisable en `localhost` mais n'est pas disponible en ligne.

VI. Pistes d'amélioration

Plusieurs pistes d'amélioration sont envisagées pour ce projet.

A. Amélioration du modèle

Le modèle créé n'est pas suffisamment entraîné pour ne faire aucune erreur. De ce fait, il pourrait être intéressant d'augmenter le nombre de données, soit par l'ajout manuel de nouveaux morceaux de 30 secondes, soit en effectuant une nouvelle *data augmentation*, en modifiant certaines valeurs relatives à la musique, comme la fréquence par exemple.

De plus, l'entraînement sur d'autres types d'architecture que les réseaux de convolution est à penser. Pour cela il faut plonger dans la littérature de façon plus poussée afin de savoir si c'est possible et comment.

B. Le site web

Une des améliorations voulues mais que je n'ai pu mettre en place par manque de temps est une modification de l'envoi des données sur le serveur. En effet, en l'état actuel du site web, le fichier audio est envoyé au serveur.

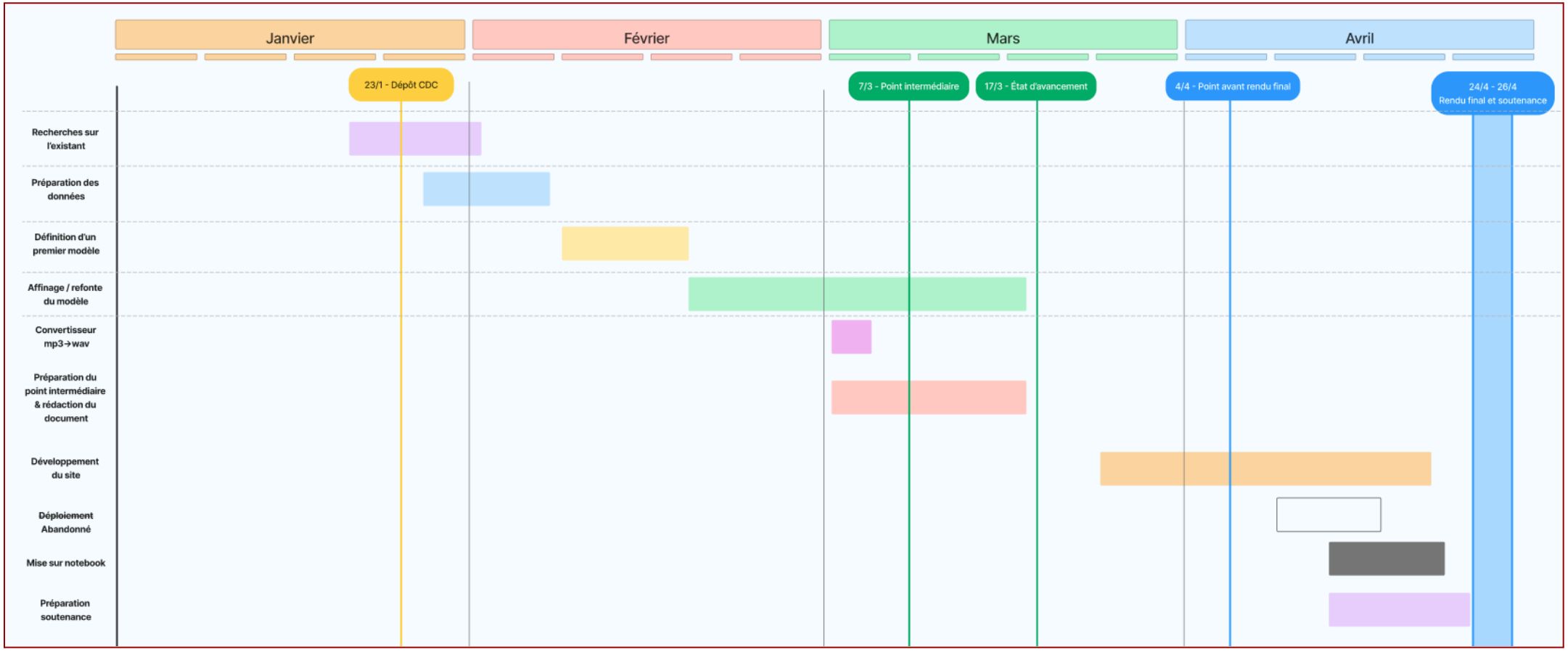
Cependant, ceci n'est pas très habile sur le plan sécurité du site. Il faudrait donc pouvoir récupérer les MFCC du fichier audio directement côté client afin de n'envoyer au serveur que le fichier texte les contenant.

Pour pouvoir effectuer de telles opérations côté client en python, une librairie existe, nommée **Pyodide**.

Dans l'éventualité où je pourrai déployer le site web comme voulu initialement, je serai dans l'obligation de faire ces changements.

VII. Planning final

Voici le planning prévisionnel revu selon l'état d'avancement actuel, pour la seconde partie du projet informatique.



VIII. Bilan personnel

Le Projet Informatique Individuel m'a permis de faire un premier pas dans le monde de l'intelligence artificielle, parcours que j'aimerais poursuivre en 3ème année.

De ce fait, j'ai pu comprendre la façon dont fonctionne un réseau de neurones ainsi qu'en développer un, et le manipuler de sorte à mesurer l'impact des divers changements dans le traitement des données mises en entrée.
Au sujet desdites données, j'ai pu peser l'importance du nombre et de la qualité de ces dernières dans un projet de machine learning.

En outre, j'ai trouvé très agréable d'avoir l'opportunité d'entreprendre le projet de notre choix, et d'avoir des heures consacrées à celui-ci. En effet, j'ai l'habitude de faire des projets en dehors des heures de cours, mais avoir un suivi par un tuteur et des retours par des professeurs est plus motivant pour se plier à des dates butoirs et avoir des retours constructifs.