Rapport PLDAC

2018-2019

Classification d'albums musicaux à partir de traces de diffusion spatio temporelle

Fait par :
Abdelraouf KESKES
Tony GOSSE-DUMESNIL

Encadrant :
Sylvain LAMPRIER

SOMMAIRE

1. Introduction

La plateforme de streaming musical Deezer utilise les genres musicaux pour son système de recommandation. Cependant, beaucoup d'albums diffusés sur Deezer ne sont pas catégorisés et n'ont aucune étiquette de genre musical . Notre problème est donc de faire une classification de genre de ces musiques en utilisant le deep learning.
Une des solutions les plus précises est de faire de l'apprentissage à partir de données audio. Or, ce processus d'apprentissage est très coûteux en mémoire et en temps et itérer ce processus sur les données non classées parmi les 100 millions de titres que contient la base de données de Deezer est juste fastidieux .

Dans notre projet nous proposons donc une nouvelle approche pour attaquer ce problème. Nous allons apprendre à superviser les genres musicaux en se basant sur les traces de diffusion spatio-temporelles des albums comme données d'entrée . Nous allons donc explorer l'aspect spatial qui est la ville de la trace de diffusion ainsi que l'aspect temporel qui sera la date de la trace de diffusion pour pouvoir en tirer une connaissance sur le genre musical.

Vous trouverez dans ces liens le code source complet du projet :

Google Drive

https://drive.google.com/drive/folders/1KdQUquR-WYDZXNpRh7cQV7s-YAnyBkKP?usp=sharing

Github

https://github.com/raoufkeskes/Deezer_MultiLabel_MusicGenreClassification.git

2. Etat de l'art

a. Deezer et la data science

Deezer est une grande entreprise lancée en 2007 par les deux français Daniel Marhely et Jonathan Benassaya. C'est une plateforme de streaming musical comptant aujourd'hui plus de 14 millions d'utilisateurs dans plus de 180 Pays.
Cette plateforme compte aujourd'hui une base permettant un accès à plus de 53 millions de titres ainsi qu'à plus de 100 millions de playlists différentes.
Le big data joue donc un rôle prépondérant dans la vision stratégique de Deezer. L'utilisation d'algorithmes d'apprentissage permet à Deezer d'analyser un nombre toujours plus volumineux de données sur les différentes tendances de comportement utilisateur sur leur plateforme.
Afin de proposer un service de plus en plus personnalisé à leurs clients, Deezer a défini son organisation autour de trois grands axes : Produit, Technologie et Contenu.
Vous trouverez ci-dessous une liste non exhaustive des applications concrètes de la data science qui sont actuellement utilisées chez Deezer, basées sur différentes techniques (Machine learning classique, Deep learning , Business intelligence ...) :

Clustering des utilisateurs : La data science permet ainsi à Deezer de pouvoir segmenter les différentes typologies de clients via des algorithmes de type apprentissage non supervisé comme les K-MEANS afin d'améliorer l'expérience utilisateur en proposant des listes de musiques toujours plus intelligentes et de faciliter la recherche de titres sur la plateforme.
Détection d'émotions (mood) : En partant de l'hypothèse qu'il est possible d'inférer les émotions d'un utilisateur en fonction des titres qu'il choisit d'écouter, il est alors possible de proposer de nouveaux titres adaptés à cette émotion. Mais comment catégoriser une chanson en termes d'humeur ? Chez Deezer, c'est à l'aide de modèles utilisant le deep learning qui, à partir de l'audio et des paroles, estiment directement des grandeurs telles que l'excitation ou la valence, qui sont liées à l'émotion suscitée par une musique.
Extraction d'instruments : Les algorithmes de machine learning permettent également de détecter les instruments utilisés dans un titre.
Systèmes de recommandations : Afin de recommander de la musique à ses utilisateurs Deezer utilise des algorithmes de type filtrage collaboratif (collaborative filtering) ainsi que des approches basées sur le contenu (Content-based filtering). Par exemple, si Deezer me classe parmi les gens qui aiment le classique ainsi que le jazz, alors il me proposera des chansons que les autres utilisateurs de ce groupe aiment.
Deezer ne fait pas que stocker des fichiers sons, il conserve aussi 1 To de logs chaque jour. Ces données permettent de calculer les royalties des artistes, de remonter des chiffres pour les reporting des maisons de disques, ou réaliser des analyses en interne. Ce Big Data sert également à établir les "tops", les classements des morceaux les plus écoutés, et les recommandations pour les radios thématiques personnalisées. Ce que Deezer appelle le "mix".
Business Intelligence & KPI : La création des Key Performance Indicators (KPI) a pour but d'aider à la prise de décisions vis à vis d'une fonctionnalité. Il est important de garder à l'esprit de limiter le nombre de KPIs ainsi que de dashboards pour ne pas être noyé sous les métriques. Prenons l'exemple d'une playlist mise en avant sur la page d'accueil alors nous observons que son volume d'écoutes explose, mais que la satisfaction diminue grandement également. Ceci peut s'expliquer par le fait que, du fait de sa mise en avant généralisée, la playlist devient moins ciblée . Des mesures de satisfaction pertinentes peuvent être par exemple la durée d'écoute ou le fait d'ajouter un morceau à une playlist personnelle.

b. Le machine learning au sein de la musique

Avec l'explosion exponentielle des réseaux sociaux et des plateformes de streaming , tous les artistes se sont réorientés vers ces derniers en uploadant tous leurs titres, informations personnelles et professionnelles… dans le but de moderniser leur art .

Cette dernière nous a créé avec le temps des données massives qui inévitablement se sont impliquées dans le machine learning, et récemment le Deep learning. Plusieurs chercheurs se sont donc engagés à résoudre une multitude de problèmes autour de la

musique .

En l'occurrence on a :

- La Détection de tempo, pulsation, tonalité, hauteur, accords ...
- L'identification de l'artiste
- La reconnaissance des instruments
- La classification d'humeur, sentiments...
- La génération de musique (Compositeur, DJ …)

avec toute cette masse de données et toute la puissance computationnelle qu'on a aujourd'hui , les réseaux de neurones profonds prennent petit à petit l'ampleur sur le machine learning classique sur certaines tâches.
Le camembert ci-contre représente le pourcentage des différentes applications du deep learning dans le son :

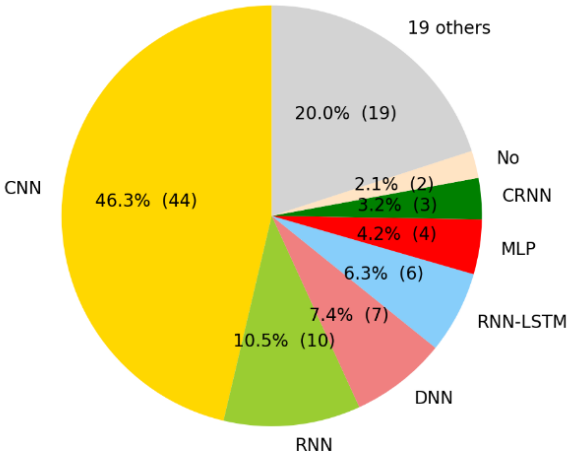


Figure 2.b.1 : Répartition des différents modèles de classification appliqués à la musique

c. Apprentissage avec des données spatio-temporelles

Les données spatio-temporelle sont présente dans beaucoup de domaines telles que l'écologie, la météorologie, la biologie, la médecine, l'économie, le trafic et la vision. Ces données peuvent provenir de multiples sources comme les images satellites, les vidéos de caméras, les coordonnées GPS, etc. Le problème de ces données est leurs très grandes tailles et le challenge est de réussir à réduire leur dimension sans perdre d'informations capitales, choses qui est très exploité en statistique et en machine learning.

Le deep learning à développé de nombreux modèles capables de capturer les informations importantes pour différentes tâches. Dans le cas des données dynamiques, les réseaux de neurones récurrents sont très adaptées pour certaines tâches comme la classification de séquences, la prédiction de séquence et la génération de séquence.
Récemment, un nouveau modèle de réseau de neurones est apparu: le "Spatio-temporal neural network (Stnn)" modélisé pour capturer les corrélations entre les multiple séries au niveau spatial et temporel.

L'apprentissage avec des données spatio-temporelles est très utilisé dans la vie quotidienne. Dans la météo on prédit la direction et la puissance de vent le lendemain ou on prédit la température à un endroit donnée pour la semaine. Dans certaines ONG, on prédit le nombre de naissance ou de mort à un endroit spécifique. Dans les entreprises routière, on prédit la circulation routière à l'heure suivante.

Exemples:

Dataset	Task	n	m	nb relations	time-step	total length	training length	#folds
Google Flu	Flu trends	29	1	1 to 3	weeks	$\approx$ 10 years	2 years	50
GHO (25 datasets)	Number of deaths	91	1	1 to 3	years	45 years	35 years	5
Wind	Wind speed and orientation	500	2	1 to 3	hours	30 days	10 days	20
PST	Temperature	2520	1	8	months	$\approx$ 33 years	10 years	15
Bejing	Traffic Prediction	5000	1	1 to 3	15 min	1 week	2 days	20

Figure 2.c.1: Statistiques sur quelques datasets. n est le nombre de séries, m est la dimension de chaque séries et time-step est la durée d'un pas de temps.

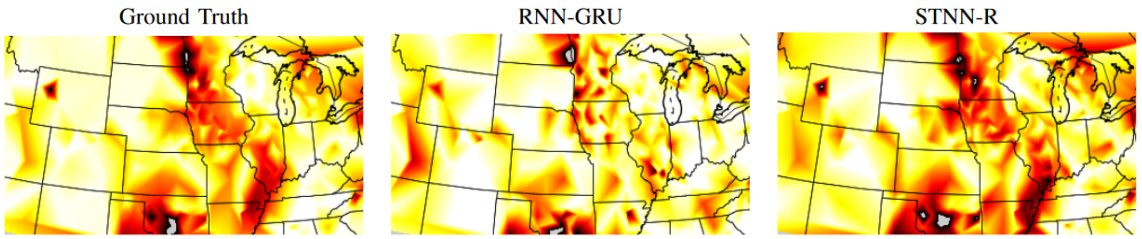


Figure 2.c.2: Prédiction de la vitesse du vent autour de 500 stations aux Etats-Unis. A gauche, la vraie vitesse du vent. Au milieu puis à droite, la prédiction au temps T+1 du vent avec le modèle RNN-GRU puis avec STNN.

d. Classification à partir des données audio

Un fichier audio est simplement une très longue série de valeurs. La fréquence d'échantillonnage classique est de 44 100Hz. Il y a donc 44 100 valeurs stockées par seconde et 2 fois plus pour les musiques en stéréo. Donc 3 minutes de musique en stéréo contiennent 7 938 000 valeurs, ce qui est énorme.
Ainsi, il faut éliminer des informations pour en retenir l'essentiel et qui sera discriminant dans la classification.

On peut déjà convertir la musique en mono pour diviser par 2 le nombre de valeurs. On applique ensuite la transformée de Fourier pour convertir nos données audio en domaine de fréquence. On l'exporte ensuite en spectrogramme avec 128 niveaux de fréquences différentes et on obtiendra un fichier PNG contenant l'évolution des fréquences de notre musique en fonction du temps. Nous prenons ensuite une résolution de 50 pixels par seconde. On découpe notre image en parts égales de 2.56 secondes.

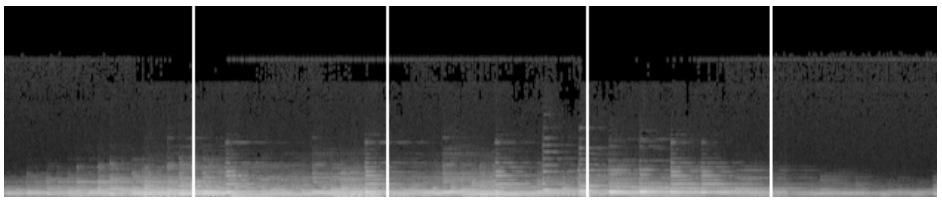


Figure 2.c.1 : Spectrogramme de la musique séparée en plusieurs parties

Après avoir découpé nos musiques en différentes images spectrales, nous avons un dataset qui contient des milliers d'échantillons pour chaque genre.
Nous pouvons maintenant utiliser un réseau de neurones convolutif profond pour classer ces échantillons. Ici on peut utiliser TFLearn de Tensorflow.

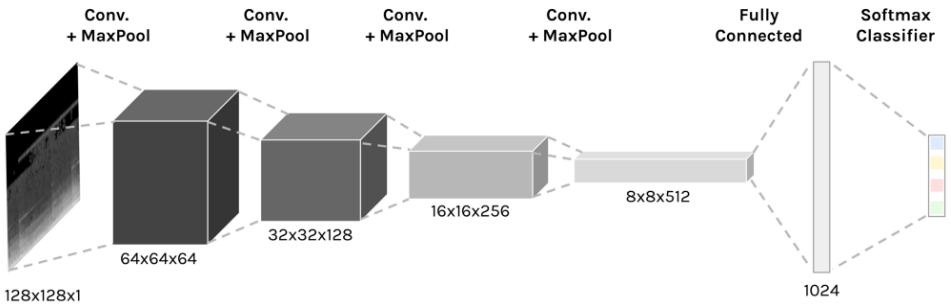


Figure 2.c.2 : Schéma du modèle de réseau de neurones convolutionnel pour classifier un spectrogramme de musique

Jusqu'à maintenant nous classifions seulement les échantillons d'une musique. Pour trouver le genre final d'une musique on utilise un système de vote sur ses échantillons et le genre qui obtient le plus de vote est le genre final.

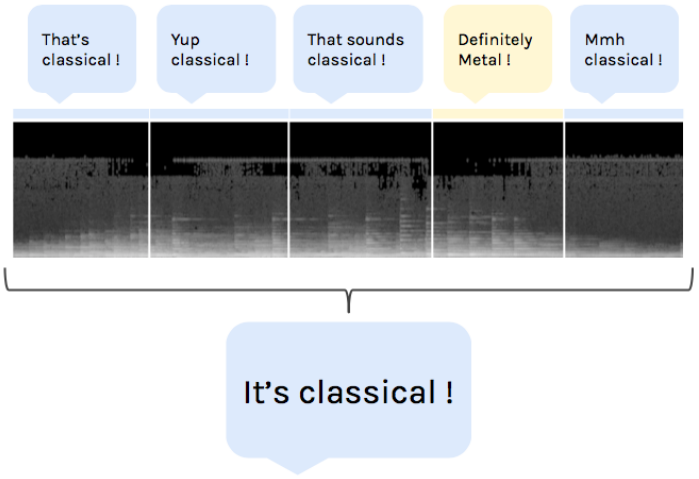


Figure 2.c.3 : Système de vote majoritaire sur les différentes parties du spectrogramme

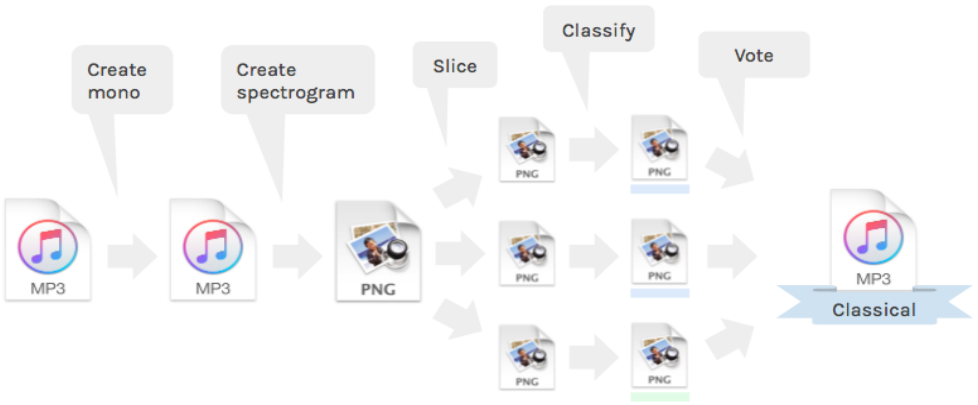


Figure 2.c.5 : Schéma global du modèle de classification supervisée de données audio

De manière générale la classification avec les signaux musicaux atteint les 61% de précision dans la classification de 10 genres multi classes. Mais ce modèle utilisant des réseaux de neurones à convolution et le système de vote en plusieurs segments du signal musical est allé jusqu'à 70% de précision avec 15 genres. Cette précision est celle atteinte par un humain.

3. Analyse des données

a. Les données

Les données ont été récupérées dans un format JSON et ont été transformées en dataframes (pandas) pour l'analyse statistique. La figure ci-dessous montre ces dataframes :

	age_group	album_id	d	loc_city	nstreams	nusers	platform_name
0	18-24	14171536	20161031	Paris	24	14	ios
1	18-24	14171536	20170427	Paris	7	5	ios
2	unk	14171536	20161008	Paris	10	6	android
3	18-24	14171536	20161220	Levallois-perret	5	3	ios
4	18-24	14171536	20170929	Strasbourg	5	3	ios

	album_id	genre_name
0	14171536	Rap/Hip Hop
1	14171536	Rap français
2	14246338	Pop
3	14400142	Dance
4	14419282	Rap/Hip Hop

Figure 3.a.1 : Les données récupérées sous le format DataFrame de pandas

b. Quelques chiffres

- 1557 albums dont seulement 1543 étiquetés.
- 2 348 509 traces de diffusion.
- 2683 villes internationales mais majoritairement françaises.
- 48% des données proviennent de Paris.
- 84% du streaming vient de Paris.
- 39% des auditeurs sont des jeunes de 18 à 24 ans.
- 1er enregistrement le 7 octobre 2016.
- Dernier enregistrement le 29 septembre 2017.
- 60 jours de données streaming pour un album au minimum à disposition.
- 358 jours de données streaming pour un album au maximum à disposition
  Environ 1 an de données de diffusion spatio-temporelle.
- La date de sortie la plus récente d'un album enregistré dans la base est le 28 juillet 2017.
- 51 genres musicaux.
- Nombre de valeurs manquantes dans la colonne âge est 351 390.

c. Statistiques sur les genres musicaux

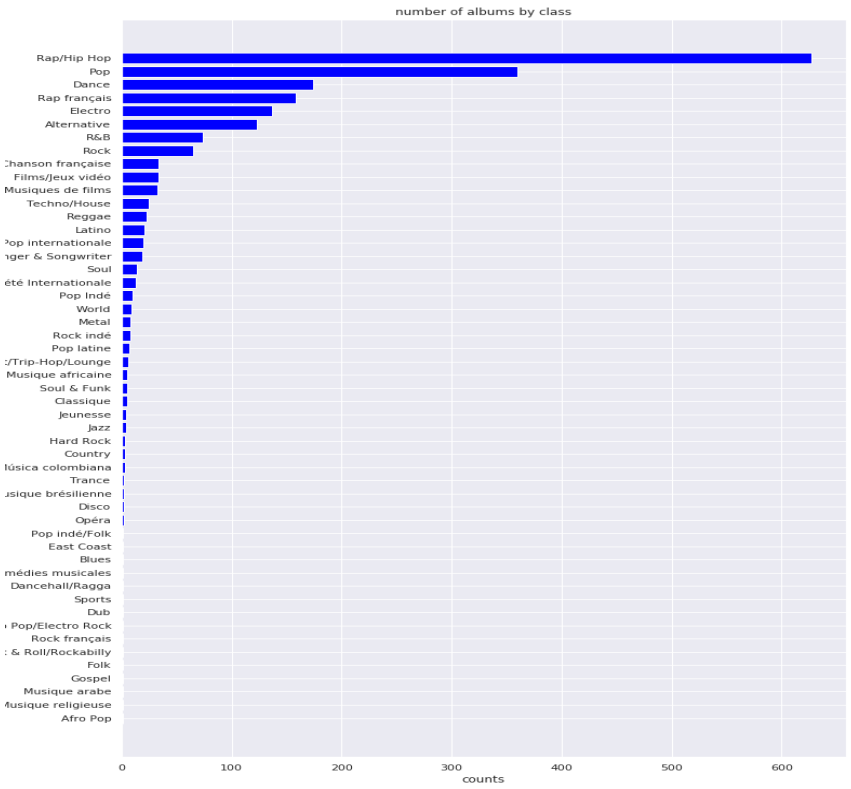


Figure 3.c.1 : Distribution du nombre d'albums par genre

Nous remarquons que nous sommes face à un problème ayant une distribution déséquilibrée des données entre les différentes classes (Unbalanced data). Le genre "Rap / Hip Hop" prédomine le dataset avec 627 albums. Cependant, 15 genres musicaux ont seulement un seul album dont plusieurs genres sont en commun sur un même album.

Hypothèse 1:
Nous décidons donc de supprimer tous les genres ayant moins de 5 albums, car nous jugeons que ces classes sont trop pauvres en données.

Résultat :
24 genres ont été supprimés engendrant la suppression de seulement 6 albums.
Ce qui nous laisse au total 1537 albums finaux.

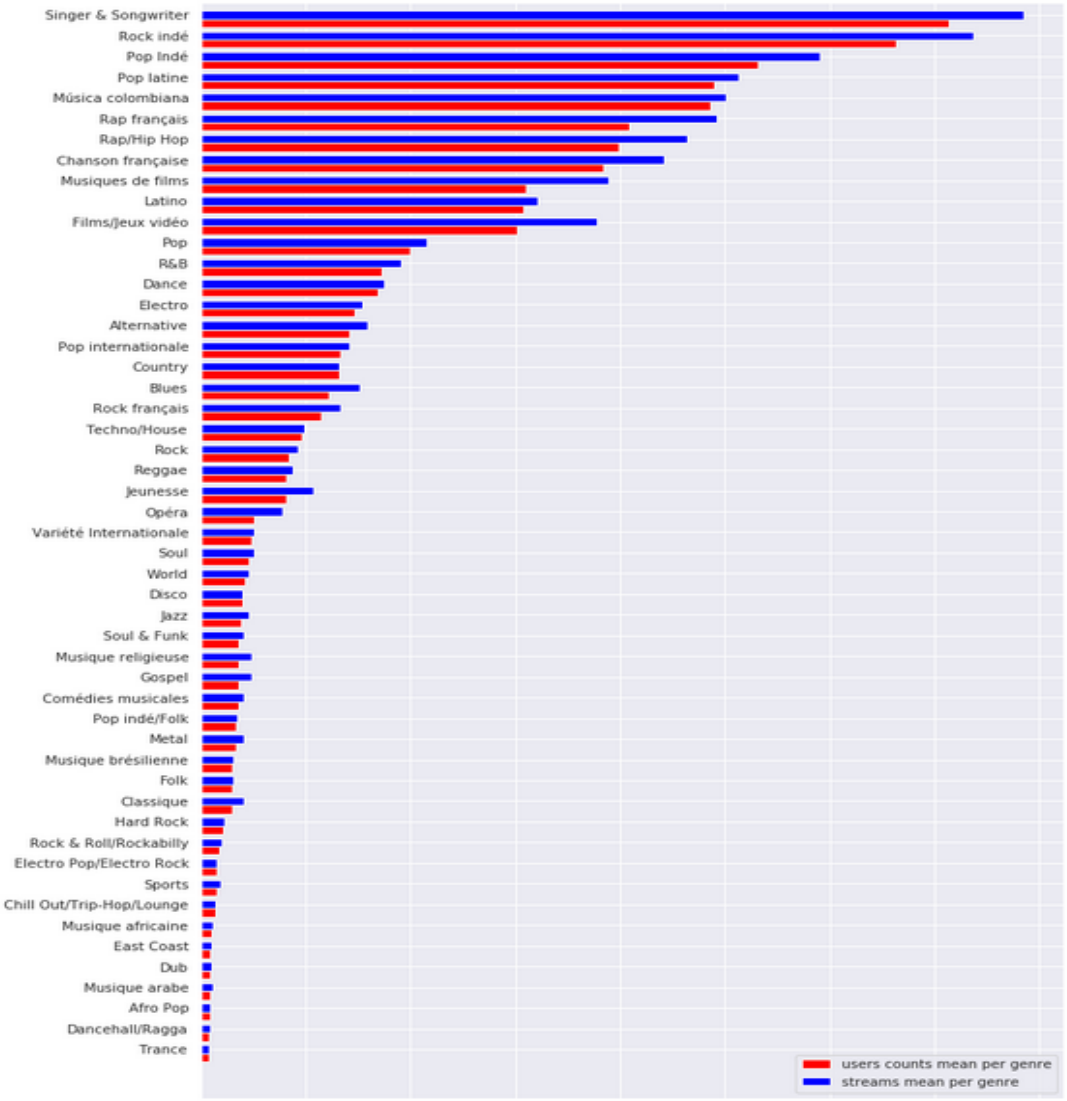


Figure 3.c.2 : Distribution du nombre de streaming moyen et le nombre d'utilisateurs moyen par genre

Nous constatons que le genre "Singer & Songwriter" est celui dont les albums ont le plus d'écoutes streaming en moyenne, suivi par les genres "Rock indépendant" , "Pop indépendante" etc...
En effet, les albums et singles de ces genres font partie des plus vus et plus célèbres sur les plateformes de musique et sur Youtube. Dans les dernières positions, on retrouve l'Afro pop, le Dancehall et la Trance.

Nous remarquons qu'en moyenne le nombre d'utilisateurs ainsi que le nombre de streaming ont la même distribution entre les différents genres.

Hypothèse 2 :
Nous décidons donc de ne garder qu'une seule information entre "**nusers**" et "**nstreams**". Nous avons opté pour "**nusers**" qui nous semble intuitivement plus stable que "**nstreams**", car par exemple un utilisateur peut streamer plusieurs fois le même album le même jour dans la même ville.

d. Statistiques sur la population

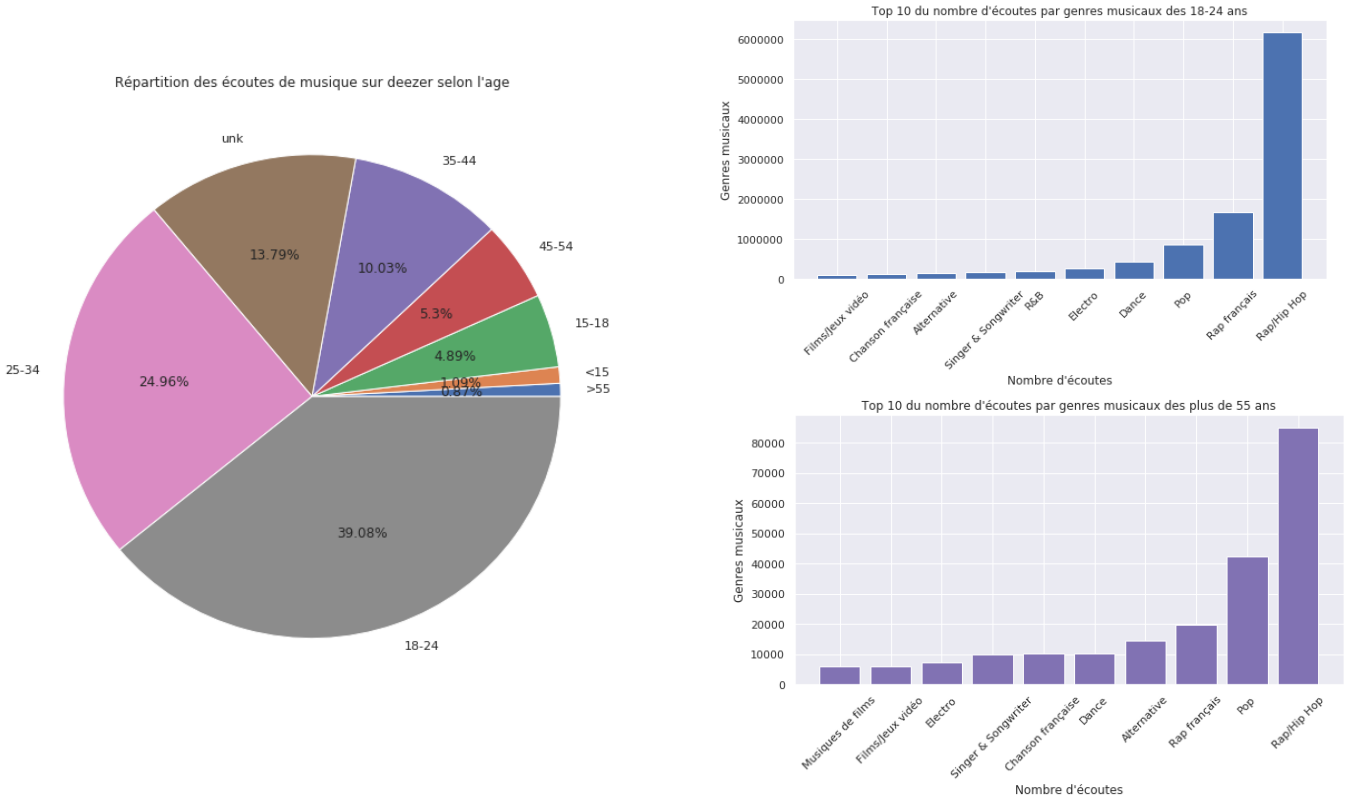


Figure 3.d.1: Répartition des écoutes sur Deezer selon l'âge de l'auditeur,
Top 10 des genres écoutés par les 18-24 ans,
Top 10 des genres écoutés par les plus de 55 ans

Les 18-24 ans représentent les auditeurs les plus nombreux avec 39% des écoutes sur Deezer, devant les 25-34 ans avec 25%.
Les auditeurs en dessous de 15 ans et au dessus de 55 ans sont les moins présents sur Deezer. Ceci s'explique par une plus faible utilisation des smartphones par cette population.
Parmi les 18-24 ans, le genre le plus écouté est bien évidemment le Rap/Hip-Hop suivi du Rap français et de la Pop. Pour les plus de 55 ans, le schéma est un petit peu différent, Le Rap/Hip-Hop est toujours en tête mais la Pop arrive avant le Rap français. Sinon on ne constate pas de grande différence sur les préférences de genres selon les âges même avec un grand écart d'âge. Notons aussi que nous avons 351 390 traces de diffusion avec catégorie "unknown" ce qui signifie qu'elles sont manquantes .

Hypothèse 3 :
Nous décidons donc de supprimer la dimension âge que nous jugeons non pertinente, où plus exactement non discriminante .

e. Statistiques sur les villes

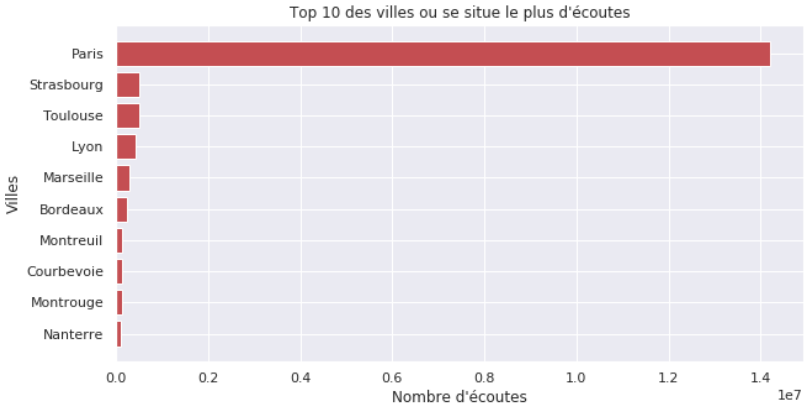


Figure 3.e.1 : Le nombre de streaming par ville

Nous constatons que le top 10 des villes qui écoutent le plus de musique sur Deezer sont françaises. Cela se justifie par le fait que Deezer est une compagnie française légèrement commercialisée à l'échelle internationale face aux concurrents comme Spotify, Youtube Music, Apple Music, SoundCloud. Notons aussi que la majorité des écoutes provient de la région parisienne .

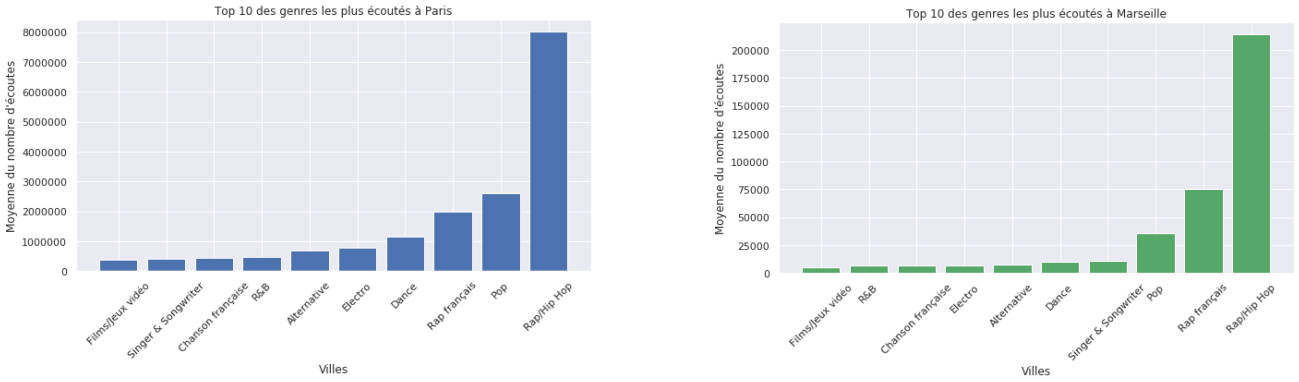


Figure 3.e.2: Top 10 des genres à Paris et top 10 des genres à Marseille

Malgré le déséquilibre entre les classes , le classement des genres les plus écoutés à Paris et à Marseille diffère beaucoup. Le “Rap Français” arrive en deuxième position suivi par le “Pop” alors qu’à Marseille c’est l’inverses. Ce classement nous montre que la dimension spatiale est importante pour la classification.

Autres Hypothèses :
La colonne “platform_name”, qui représente le système d’exploitation des auditeurs, prend comme valeurs : ['ios', 'android', 'web', 'osx', 'windowsphone', 'windowsuniversal'].

Nous avons décidé de supprimer cette dimension qui intuitivement ne nous semble pas discriminante pour la détection des genres musicaux.

Note:

1) Dans la partie qui suit , nous tenons à vous informer que le projet a été développé majoritairement sur la plateforme Google Collab dédiée au "Machine Learning" et plus précisément au "Deep Learning ".Ils proposent une machine puissante avec les caractéristiques suivantes:

- GPU: Tesla T4 compute 7 , having 2560 CUDA cores , 16GB GDDR5 VRAM
- CPU: 1xsingle core hyper threaded Xeon Processors @2.3Ghz i.e(1 core, 2 threads)
- RAM: ~12.6 GB Available
- Disk: ~33 GB Available

Ainsi , toute information qui sera relative au matériel , elle sera basée sur "**Google Collab Platform**"

2) Dans la partie **protocol expérimental** , aucune figure ou résultat ne sera commentée ou discutée .Tout sera détaillé dans la partie qui la suit , en l'occurrence **" Discussion des résultats " .**

4. Protocol Expérimental

a. Modèle de données

i. Données d'apprentissage

Dans tout ce qui suit , tout ce que nous allons appelé "Step" fera référence à une date dans le dataset :

- Step = 0 => Date de sortie de l'album
- Step = 357 => taille de fenêtre temporelle maximale vu que les données ne dépasseront jamais cette borne de 358 jours de diffusion comme mentionné dans la partie **3.b**.

Le "nbSteps" fait référence au nombre de Step agrégé .

Exemple:

un album de taille (358 , 2683) a un nbSteps = 358 , peut se transformer à un album de taille (4 , 2683) avec un nbSteps = 4 , en agrégeant par somme .

Il ne faut pas confondre aussi Step qui est un indice allant de 0 à 357 et le nbSteps qui est un nombre allant de 1 à 358 .

Nous démarrons initialement du DataFrame suivant préprocessé selon les hypothèses citées précédemment :

album_id	date	loc_city	nusers
13972552	2016-11-03	Paris	3
13972552	2016-11-19	Paris	3
13972552	2016-11-20	Paris	3
13972552	2016-11-24	Paris	3
13972552	2016-12-09	Paris	3

Figure 4.a.i.1 : DataFrame des données d'apprentissage en le transformant en un tenseur de 3 dimensions de taille 1537 albums x nbSteps x 2683 villes

$$X = \begin{bmatrix} \begin{bmatrix} 1 & 2 & 0 & . & . & . & . & 4 \\ . & . & . & . & . & . & . & . \\ 3 & 0 & 1 & . & . & . & . & 0 \end{bmatrix} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \begin{bmatrix} 7 & 2 & 0 & . & . & . & . & 4 \\ . & . & . & . & . & . & . & . \\ 0 & 0 & & . & . & . & . & 0 \end{bmatrix} \end{bmatrix}$$

où :

- **X** est un tenseur qui représente le dataset final entier .
- Chaque $\mathbf{X}_{\mathbf{i}}$ est une matrice représentant un album de taille (nbSteps x 2683 villes).
- **nbSteps** ∈[1 , 358]
  - ➢ C'est un paramètre de représentation à tuner .
  - ➢ le 1 représente le fait de ne pas exploiter l'aspect temporel et de grouper tous les jours de diffusion sur une même ville en une seule information.
  - ➢ le 358 représente l'exploitation maximale de l'aspect temporel, car c'est le nombre de jours maximal de données de diffusion pour un album.

- L'information contenue dans le tenseur représente le nombre d'utilisateurs noté "**nusers**" ayant écouté un album **i** dans **une ville** à une **date=step** ∈**[0 , 357]**

Visualisation des données :

Figure 4.a.i.1 : Exemple de 2 classes montrant l'astuce de visualisation multi label

Nous avons dans cette exemple 2 classes et l'astuce de visualisation est d'utiliser des "markers cercles" avec de différentes tailles et couleurs .

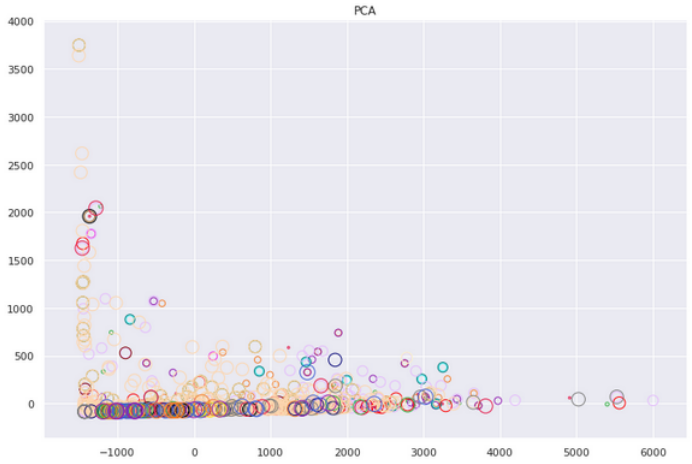


Figure 4.a.i.2 : Visualisation des albums en PCA avec nbSteps = 1

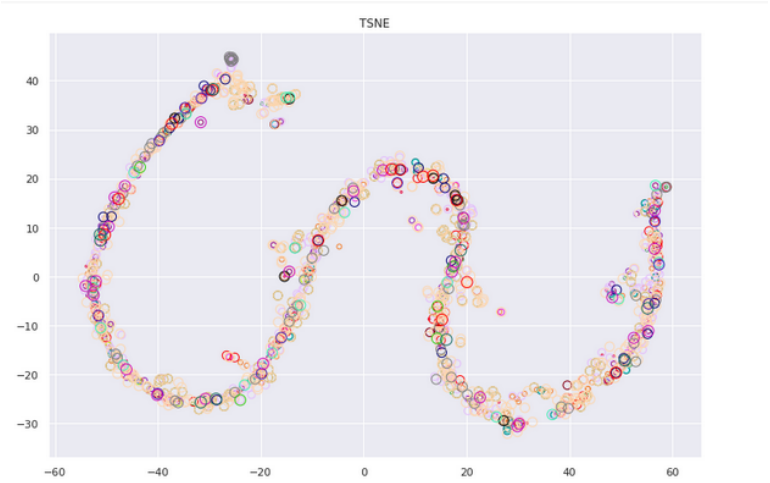


Figure 4.a.i.3 : Visualisation des albums en TSNE avec nbSteps = 1

On remarque que sur les visualisations PCA et TSNE la formations de clusters mais assez dispersé notamment pour la visualtisation. On peut en déduire les genres sont un peu séparable même avec nbSteps=1, c'est- à-dire en prenant en compte seulement l'aspect spatial.

ii. Les Labels

Vu que notre problème est un problème "Multi-Labels" nous avons peu de manières de représenter les Labels et nous avons eu comme choix :

- **One-hot Encoding:** Il crée une colonne pour chaque valeur. Une ligne reçoit un 1 si la ligne contenait la valeur de cette colonne et un 0 si elle ne l'était pas.Dans notre cas ça sera donc 27 colonnes.
  Comme notre problème est de type "Multi-label" , l'encoding est censé être nommé "Multi Hot encoding" .Mais , par abus de langage nous ils l'appelent aussi "one hot encoding " .

- **Binary Encoding**: Il utilise un codage binaire pour chaque valeur. Dans notre cas nous avons 27 labels < 32 . Ils peuvent donc être codés avec 5 colonnes ($32=2^5$) .

- **BaseN Encoding**: C'est une généralisation des encoding précédents. Il est très rarement utilisé en pratique
  - ➢ Si N = 1 , alors c'est du One-hot .
  - ➢ Si N = 2 , alors c'est du binary .
  - ➢ etc …

- **Autres : Contrast Encoders , Bayesian Encoders , etc …**

Nous avons opté pour le **One-hot Encoding,** car c'est le plus utilisé dans les articles scientifiques. De plus, il est simple, interprétable et très adapté aux réseaux de neurones. Comme le montre la matrice suivante :

$$Y = \begin{bmatrix} 1 & 0 & 0 & . & . & . & . & 1 & 0 & 1 \\ 0 & 1 & 1 & . & . & . & . & 0 & 0 & 1 \\ . & . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & . & . & 1 & 0 & 0 \end{bmatrix}$$

où Y est une matrice 2D de taille (1537 albums x 27 genres)

b. Métriques d'évaluation utilisées

i. Mesures basées sur les Labels

Ce sont des Métriques inspirées du domaine de la recherche d'information .
Notations :

- TP ⇔ True Positive , FP ⇔ False Positive , FN ⇔ False Negative
- ci est une classe , C est l'ensemble de toutes les classes et D est le dataset
- k est l'ordre des métriques Précision , Rappel qui sera utilisé que pour la précision pour le calcul de la AvgP (Average Precision) et pour la MAP (Mean Average Precision)

Précision , Rappel et F1-mesure (Macro ou Micro) :

. Macroaveraging Precision $Prc^{macro}(D) = \frac{\sum_{c_i \in C} Prc(D, c_i)}{|C|}$

. Macroaveraging Recall $Rec^{macro}(D) = \frac{\sum_{c_i \in C} Rcl(D, c_i)}{|C|}$

. Microaveraging Precision $Prc^{micro}(D) = \frac{\sum_{c_i \in C} TPs(c_i)}{\sum_{c_i \in C} TPs(c_i) + FPs(c_i)}$

. Microaveraging Recall $Rcl^{micro}(D) = \frac{\sum_{c_i \in C} TPs(c_i)}{\sum_{c_i \in C} TPs(c_i) + FNs(c_i)}$

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Average Precision (AP) and Mean Average Precision (MAP) :

$$Ytrue = \begin{bmatrix} 1 & 1 & 0 & . & . & . & 1 & 0 \end{bmatrix}$$

$$YPred = \begin{bmatrix} 0.1 & 0.8 & 0.001 & . & . & . & 0.88 & 0.02 \end{bmatrix}$$

$$AP(c_i) = \frac{1}{|K|} \sum_{k=1}^{27} Precision@k(ci)$$

$$MAP = \frac{1}{|C|} \sum_{c_i=1}^{27} AP(c_i)$$

où

- K est le rang qui servira de seuil de coupure pour le calcul de la Précision@k variant bien sûr de 1 à 27.
- Pour le calcul de la Précision@k , nous aurons besoin aussi de trier les classes selon les valeurs des Y_pred retournées dans un ordre décroissant.

---

ii. Mesures basées sur les Exemples

Hamming Loss :

$$\frac{1}{|N| \cdot |L|} \sum_{i=1}^{|N|} \sum_{j=1}^{|L|} \operatorname{xor}(y_{i,j}, z_{i,j}), \text{ where } y_{i,j} \text{ is the target and } z_{i,j} \text{ is the prediction.}$$

où N est le nombre d'exemples et L est le nombre de labels = 27 .
Cette métrique n'est pas très pertinente, car elle compte le nombre de fautes sur notre matrice Y_pred par Rapport à la vraie Matrice Y. Comme c'est une matrice sparse les zéros prédits qui sont gérés par une variable "Threshold" seront considérés comme une bonne réponse, ce qui n'est pas très bon pour l'évaluation. Exemple illustratif **:**

$$Y = \begin{bmatrix} 0 & 0 & 0 & . & . & . & . & 1 & 0 & 0 \\ 0 & 1 & 0 & . & . & . & . & 0 & 0 & 0 \end{bmatrix} \quad Y_{pred} = \begin{bmatrix} 0 & 0 & 1 & . & . & . & . & 0 & 0 & 0 \\ 1 & 0 & 0 & . & . & . & . & 0 & 0 & 0 \end{bmatrix}$$

Dans ce cas là, le modèle a de très mauvais résultats sur 2 albums ayant un seul vrai Label. Il a donné 2 mauvaises réponses. Pourtant, la hamming loss est égale à (4/(2*27)) = 0.07. Cette valeur nous paraît naturellement négligeable et donne de bonnes impressions sur le modèle.

Hamming score :

souvent noté accuracy (Example-Based) :

$$A = \frac{1}{n} \sum_{i=1}^{n} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

où N est le nombre d'exemples , Yi est le vrai vecteur de labels pour un album i et Zi est le vecteur de labels prédit par le modèle .

Soft Accuracy :

notre métrique personnalisée qui pour chaque album compte le nombre de labels bien prédits à 1 sur le nombre total de labels attendus à 1. C'est une sorte de Rappel Example-based:

$$SoftAccuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{j=27} I(y_j = ypred_j = 1)}{\sum_{j=1}^{j=27} I(y_j = 1)}$$

où N est le nombre d'exemples et **I** est la fonction indicatrice.
L'idée de la métrique se base sur 2 aspects :

-Vu que les zéros bien prédits seront ajustés avec un paramètre "**Threshold**" sur la sortie du modèle, nous n'avons donc pas trop envie de les inclure dans l'évaluation. Le paramètre threshold est une valeur de seuil qui permet de garder la classe si la valeur de prédiction dépasse celle-ci.
-Ce qui nous importe le plus c'est de prédire quelques bons genres, même si on rajoute un ou deux mauvais genres avec, ça ne sera pas trop grave.

Exemple :
threshold = 0.3
Labels = ["Rap/Hip-Hop", "Rock indie", "Techno", "Trance", "Rap Français", "Rock, "Pop"]
YTrue = [1,0,0,0,1,0,1]
YPred = [0.45, 0.2 , 0.05 , 0.08 , 0.36 , 0.31 , 0.29]

Alors les labels prédit sont le "Rap/Hip-Hop", le "Rap Français" et le "Rock".
Sauf que les vrais genres de l'album sont "Rap/Hip-Hop", le "Rap Français" et la "Pop".

SoftAccuracy = 2/3 pour cet album.

Cette métrique reste trompeuse aussi si on prédit tous les genres pour tous les albums , elle donnera un résultat = 1 .

Exact Match Ratio :
C'est la métrique la plus stricte qui compte le nombre de YPred égale exactement au YTrue.

$$ExactMatchRatio, MR = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i)$$

Confusion Matrix :
Dans le multi-label, nous allons faire une approche binaire One VS All .

c. Modèles de réseau de neurones

i. Réseaux de neurones fully connected

Comme le titre l'indique, nous avons utilisé un réseau de neurones "fully-connected" ayant comme architecture :

- ➢ Input layer : nbSteps*2683 -> Relu
- ➢ Hidden Layer : nbSteps*100 -> Relu
- ➢ Output : 27

et ayant comme hyper-paramètres :

- batch size = 20
- learning rate = 0.001
- Optimizer = Adam
- Loss Function : BCEWithLogits (aka Binary Cross-Entropy with Sigmoid) proposé par Pytorch pour la classification Multi-Label .
  $$(y \cdot \ln(\text{sigmoid}(\text{logits})) + (1 - y) \cdot \ln(1 - \text{sigmoid}(\text{logits})))$$
  où y est le vrai label et logits est la sortie du reseau neurones
- epochs = 50
- Threshold = 0.2 appliqué sur le Output sigmoidé pour le transformer en un vecteur binaire qui correspond au Multi-label .Exemple : [0 1 . . . . . 1 1 0]

Ces paramètres ont été soigneusement choisis après plusieurs tests et tentatives. Nous éviterons de mettre ça sur le rapport pour éviter de l'encombrer.

Pour la représentation, nous avons commencé par tout agréger en sommant sur les différentes villes. Nous nous sommes donc fixés :

- nbSteps = 1
- X de taille 1537 x 1 x 2683

Note:

- ➢ Toutes les opérations entre tensors se passent en GPU lors de l'apprentissage et le test pour booster les calculs .
- ➢ Tous les résultats ont été arrondis à deux décimales après la virgule.

Tout d'abord, nous avons commencé par entraîner le réseau de neurones et le tester sur l'ensemble entier d'apprentissage des 1537 albums. C'était dans le but de vérifier que ce qu'on fait a du sens, pour ne pas se lancer sur une mauvaise piste.

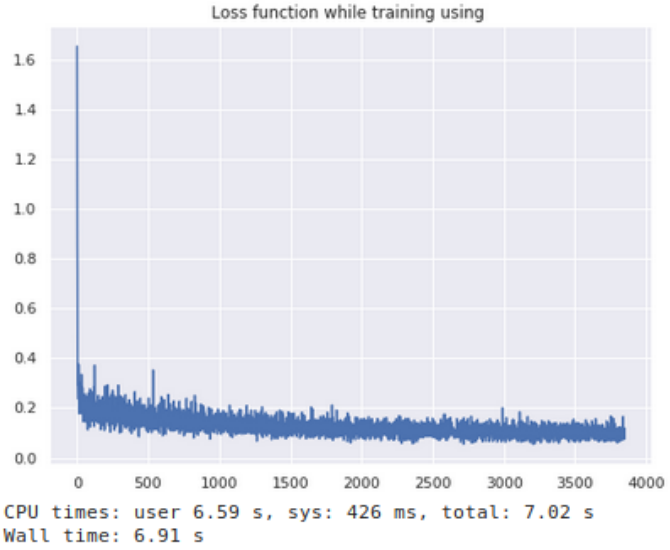


Figure 4.c.1 : La fonction Loss lors de l'optimisation du réseau de neurones

Nous avons obtenu comme résultat :

Label-Based Metric	résultat	Example-Based metric	résultat
MAP	0.46	**Soft Accuracy**	0.71
Précision(Macro)	0.61	**Hamming Score**	0.51
Recall(Macro)	0.38	**Exact Match**	0.32
F1(Macro)	0.41	**Hamming Loss**	0.05

Nous avons ensuite fait un split Train 80%/ Test 20%, avec toujours la même représentation c-à-d **nSteps** = 1. En gardant bien évidemment la même distribution et déséquilibre des données entre les classes du Train sur l'ensemble de Test. Nous avons obtenu comme résultats les figures et les tableaux récapitulatifs suivants :

Label-Based Metric	résultat	Example-Based Metric	résultat
MAP	0.34	**Soft Accuracy**	0.53
Précision(Macro)	0.44	**Hamming Score**	0.43
Recall(Macro)	0.24	**Exact Match**	0.28
F1(Macro)	0.27	**Hamming Loss**	0.06

	Precision (K=27)	Recall (K=27)	F1-score (K=27)	AvgP
Alternative	0.36	0.45	0.4	0.35
Chanson française	0.0	0.0	0.0	0.08
Chill Out/Trip-Hop/Lounge	0.0	0.0	0.0	0.05
Classique	0.0	0.0	0.0	0.01
Dance	0.3	0.62	0.41	0.39
Electro	0.75	0.14	0.23	0.33
Films/Jeux vidéo	0.0	0.0	0.0	0.23
Latino	0.5	0.14	0.22	0.37
Metal	0.0	0.0	0.0	0.04
Musique africaine	0.0	0.0	0.0	0.03
Musiques de films	0.0	0.0	0.0	0.15
Pop	0.39	0.33	0.36	0.44
Pop Indé	1.0	0.43	0.6	0.61
Pop internationale	0.75	0.21	0.33	0.49
Pop latine	1.0	0.5	0.67	0.5
R&B	0.38	0.23	0.29	0.33
Rap français	0.49	0.62	0.55	0.64
Rap/Hip Hop	0.54	0.98	0.7	0.82
Reggae	0.22	0.4	0.29	0.38
Rock	0.67	0.33	0.44	0.42
Rock indé	1.0	0.33	0.5	0.61
Singer & Songwriter	1.0	0.2	0.33	0.44
Soul	1.0	0.14	0.25	0.23
Soul & Funk	0.0	0.0	0.0	0.02
Techno/House	0.83	0.36	0.5	0.55
Variété Internationale	0.67	0.2	0.31	0.57
World	0.0	0.0	0.0	0.16

Figure 4.c.2 : Métriques d'évaluation Label-Based sur les 27 classes avec nbSteps=1

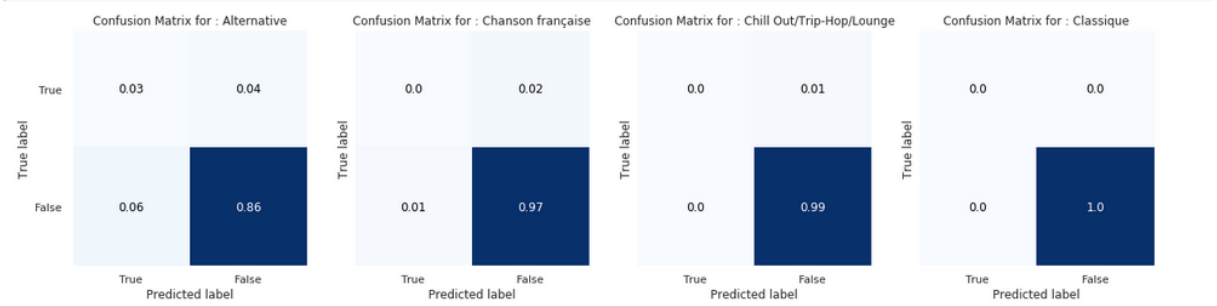
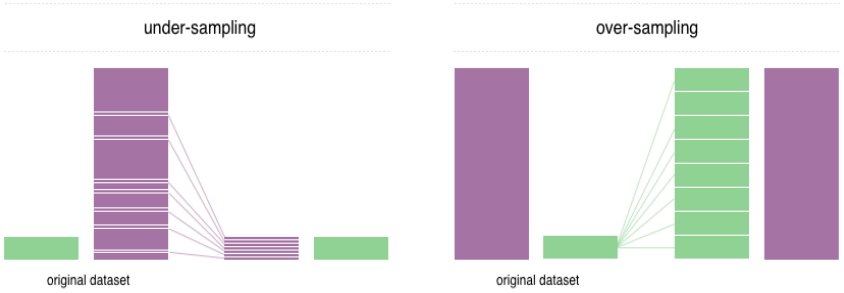


Figure 4.c.2 : Matrice de confusion (One Vs All) sur les 27 classes avec nbSteps=1

Ensuite, nous avons essayé plusieurs méthodes pour remédier au problème qu'on appelle **"Unbalanced Data"** pour espérer avoir de meilleurs résultats, voici une liste exhaustive de toutes les approches tentées qui sont parfois équivalentes :

- ➢ Suppression des données
- ➢ Réplication des données

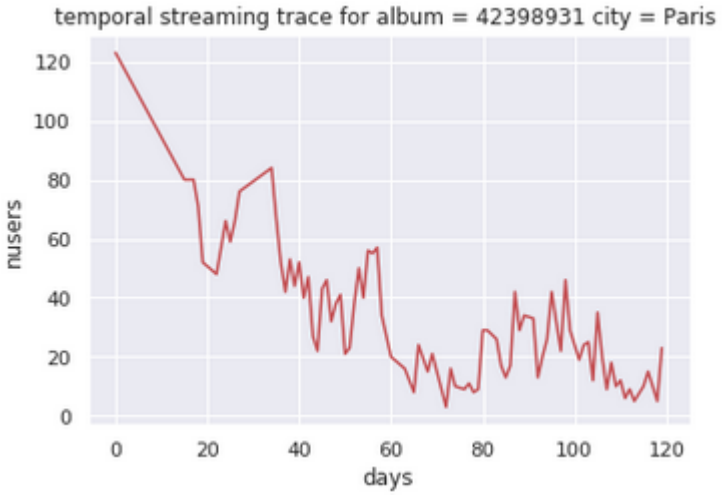


- ➢ Pondération inversement proportionnelle à la distribution des classes **de la fonction Loss** avant de rétro-propager l'erreur.
- ➢ Pondération inversement proportionnelle à la distribution des classes **des exemples Xi.**

Note :
Inversement proportionnelle => 1/count(class) + normalisation de la nouvelle distribution acquise pour sommer à 1.

Nous avons ensuite attaqué le coeur de notre projet et de notre hypothèse initiale, qui suppose que la diffusion d'albums dans l'espace et dans **le temps** peut nous fournir une information solide pour prédire son (ses) genre(s).

Voici une figure illustrant la diffusion de l'album '42398931' sur Paris , c-à-d en prenant juste une colonne (en l'occurrence "Paris") parmi les 2683 villes :



Nous avons donc fait varier le **NbSteps** entre 1 et 30, l'agrégation donc était conditionnée par NbSteps (**nbStepsMax** = 358).

Nous avons aussi tuné le paramètre **Threshold** pour voir son effet sur les différentes métriques, et ainsi de capturer le meilleur threshold possible.
Ci-dessous les différents tableaux et figures illustrant les résultats de ce processus expérimental :

Figure 4.c.3 : Evolution de la mesure MAP en fonction du **nbSteps**

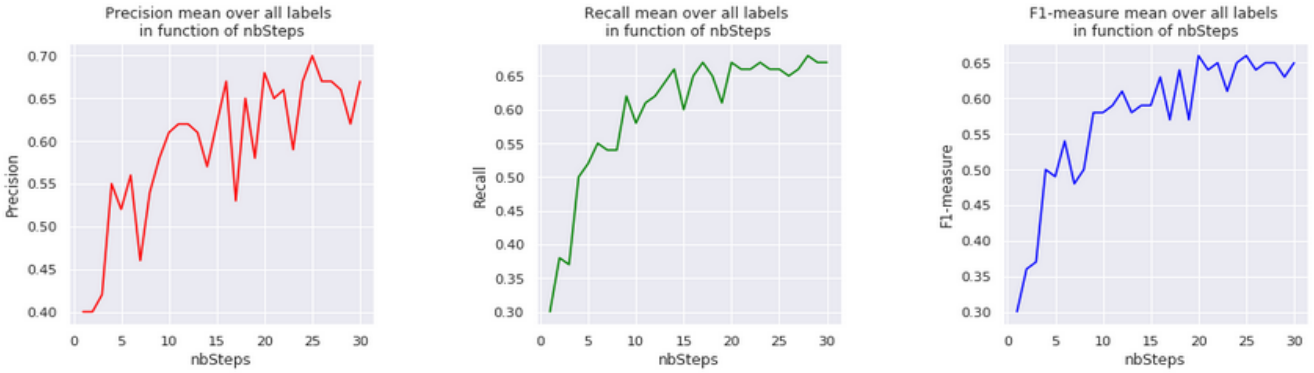


Figure 4.c.4: Evolution des Métriques Label-Based en fonction du **nbSteps**

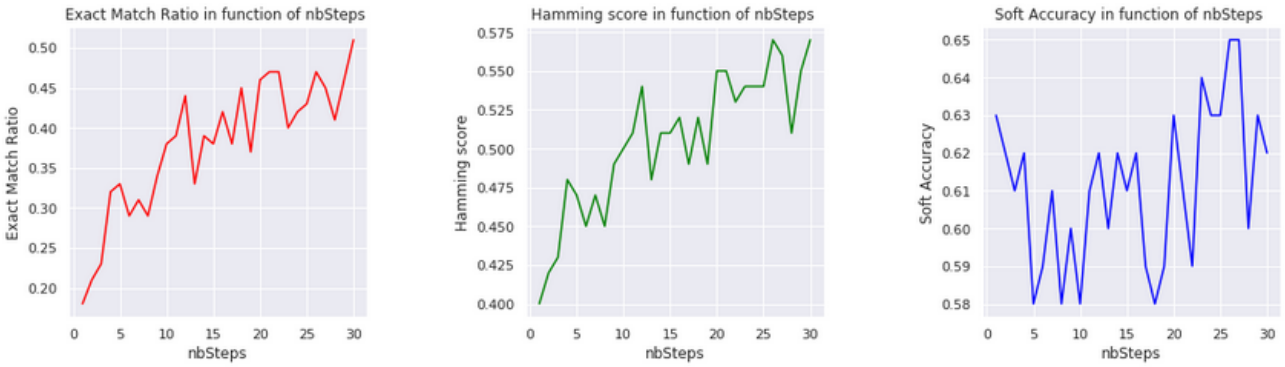


Figure 4.c.5: Evolution des Métriques Example-Based en fonction du **nbSteps**

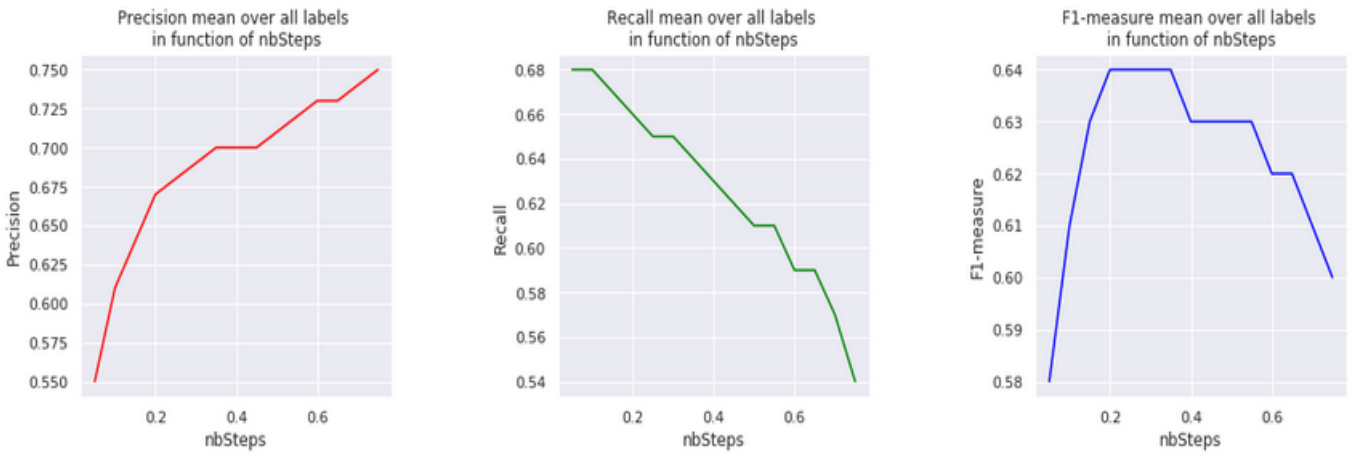


Figure 4.c.6: Evolution des Métriques Label-Based en fonction du **Threshold**

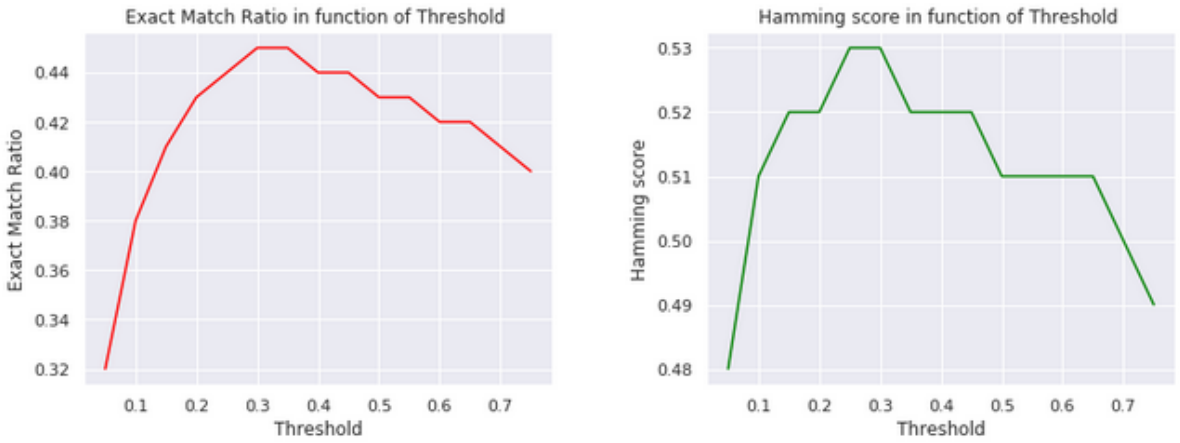


Figure 4.c.7: Evolution des Métriques Example-Based en fonction du **Threshold**

En fixant les 2 meilleures valeurs Threshold=0.3 et nbSteps=28, nous trouvons :

	Precision (K=27)	Recall (K=27)	F1-score (K=27)	AvgP
Alternative	0.5	0.52	0.51	0.52
Chanson française	0.33	0.12	0.18	0.16
Chill Out/Trip-Hop/Lounge	1.0	1.0	1.0	1.0
Classique	0.0	0.0	0.0	0.01
Dance	0.54	0.58	0.56	0.58
Electro	0.5	0.66	0.57	0.68
Films/Jeux vidéo	0.92	0.86	0.89	0.87
Latino	0.67	0.57	0.62	0.62
Metal	1.0	0.67	0.8	0.92
Musique africaine	0.0	0.0	0.0	0.01
Musiques de films	0.92	0.85	0.88	0.86
Pop	0.51	0.55	0.52	0.55
Pop Indé	1.0	1.0	1.0	1.0
Pop internationale	1.0	1.0	1.0	1.0
Pop latine	1.0	0.5	0.67	0.5
R&B	0.78	0.64	0.7	0.7
Rap français	0.78	0.85	0.81	0.88
Rap/Hip Hop	0.78	0.79	0.78	0.82
Reggae	0.62	1.0	0.77	0.77
Rock	0.68	0.72	0.7	0.73
Rock indé	1.0	1.0	1.0	1.0
Singer & Songwriter	1.0	0.8	0.89	0.8
Soul	1.0	0.71	0.83	0.73
Soul & Funk	1.0	0.5	0.67	0.52
Techno/House	0.93	1.0	0.97	1.0
Variété Internationale	1.0	1.0	1.0	1.0
World	0.0	0.0	0.0	0.03

Figure 4.c.2 : Métriques d'évaluation Label-Based sur les 27 classes avec le best nbSteps=28 et le best Threshold = 0.28

Label-Based Metric	résultat	Example-Based Metric	résultat
MAP	0.68 **(+0.34)**	**Soft Accuracy**	0.59**(+0.06)**
Précision(Macro)	0.72 **(+0.34)**	**Hamming Score**	0.53**(+0.1)**
Recall(Macro)	0.66 **(+0.42)**	**Exact Match**	0.47**(+0.19)**
F1(Macro)	0.68**(+0.41)**	**Hamming Loss**	0.03**(-0.03)**

Ces résultats engendrent donc les matrices de confusion suivantes (nous ne gardons que les matrices qui posaient problème pour voir que c'est nettement mieux) :

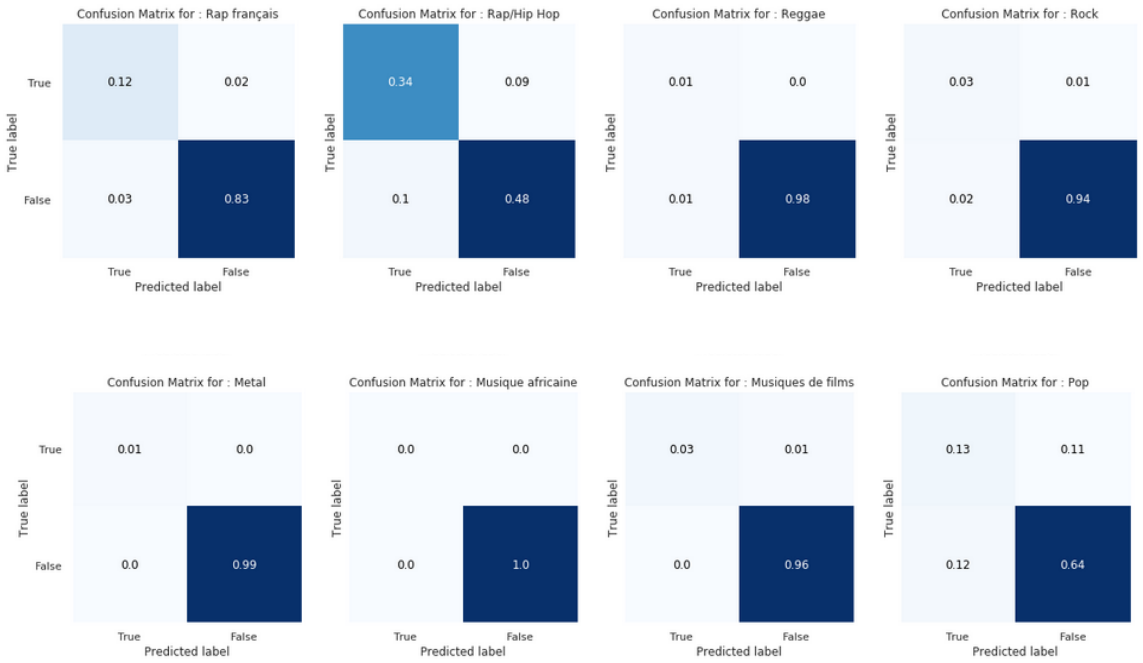


Figure 4.c.2 : Matrices de confusion pour les classes qui posaient problème avec la baseline

ii. Réseaux de neurones Convolutifs

Dans cette partie nous allons essayons de monter d'un cran et d'attaquer les données avec des réseaux convolutifs dont l'architecture est la suivante :

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 6, 86, 2669]           1,356
            Conv2d-2         [-1, 16, 41, 1332]             880
            Linear-3                [-1, 4096]      43,651,072
            Linear-4                 [-1, 128]         524,416
            Linear-5                  [-1, 27]           3,483
================================================================
Total params: 44,181,207
Trainable params: 44,181,207
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.02
Forward/backward pass size (MB): 17.21
Params size (MB): 168.54
Estimated Total Size (MB): 186.77
----------------------------------------------------------------
```

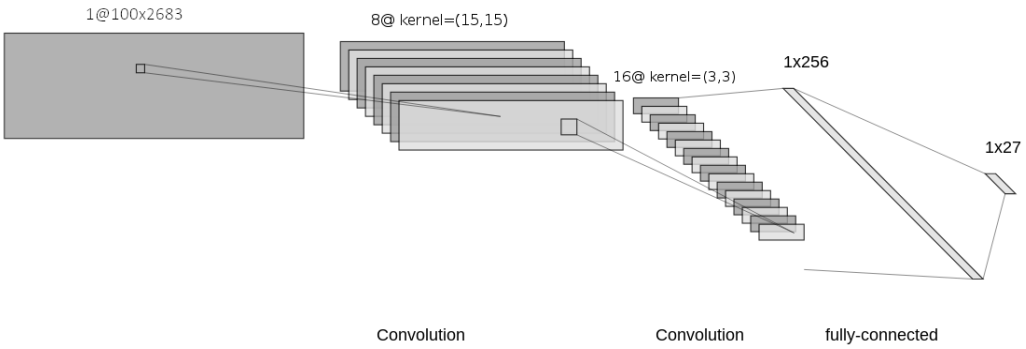


Figure 4.c.2 : Architecture du CNN utilisé dans notre problématique

Figure 4.c.1 : La fonction Loss lors de l'optimisation du réseau de neurones

Nous avons obtenu approximativement la borne de résultats atteinte avec les réseaux “fully connected“. Aucune amélioration n’a été mentionnée.

Note :
Comme les figures de chaque partie représente et véhicule généralement la même information . Nous allons regrouper le commentaire ainsi que la discussion de plusieurs figures **par partie.**

5. Discussion des résultats

a) Nous constatons que la fonction Loss du réseau de neurones s'optimise bien, en réduisant ses oscillations au fil des itérations et des époques.

b) La première expérience sur la totalité des données c-a-d les 1537 en train et en test montre que notre hypothèse spatio-temporelle a effectivement un sens.
Toutes les métriques donnent des résultats raisonnables MAP = 0.46 , Precision=0.61 , Soft accuracy = 0.71 . . .

c) Après avoir séparé nos données en Train et Test avec **nbSteps** = 1 :

- Nous réalisons qu'en ne gardant que l'aspect spatial les résultats sont loins d'être bons avec : MAP = 0.34 , Précision = 0.44 , Rappel = 0.24 . . .
- Les matrices de confusions sont prédominées par les "True False", ce qui s'explique par le fait qu'on gère ça avec un hyper-paramètre **Threshold** qui sert à transformer la sortie sigmoid en une suite de 0, 1 de taille 27 ce qui crée généralement beaucoup de zéros qui sont bons !
- Nous avons pleins de mauvais scores pour des classes pauvres en données telles que: "Chanson française" , "ChillOut/Trip-Hop/Lounge" ,"Classique","Metal" , "Musique africaine" , etc… Ceci est certainement dû au problème du "Unbalanced Data".
- Nous avons l'impression que le réseau tend vers un sur-apprentissage des genres les plus présents "Unbalanced Data". En premier lieu, nous trouvons le "Rap/Hip-Hop" avec 35% de "Faux Positif" ce qui veut dire que le modèle prédit "Rap Hip Hop" 35% de fois où il ne fallait pas le prédire.Suivi par la "Dance" ayant 19% de FP, et la "Pop" ayant 12% deFP.
- En visualisant les données avec une PCA ou TSNE , nous confirmons le fait que les labels sont très difficiles à repérer , étant donné qu'ils sont dispersés un peu partout , nous n'arrivons pas à construire des pseudo clusters interconnectés entre eux avec des points en commun entre labels . Tout paraît mélangé et aléatoire .

d) Toutes les approches pour fixer le problème du "Unbalanced Data" ont échouées et ont toutes donné de mauvais résultats. Nous justifions ça par le fait que ce déséquilibre entre les classes n'est pas un manque de données. Mais, plutôt une information voire même une connaissance qui reflète la réalité. Essayer de le rééquilibrer introduit donc un gros biais aux données qui ne ressembleront plus aux données de la réalité.

e) En augmentant la taille de la matrice représentant un album et en exploitant le temps nommé **nbSteps**, nous réalisons que les résultats et les performances du modèle augmentent en moyenne de 38% sur toutes les métriques Label-Based et en moyenne de 12% sur les métriques Example-Based avec la meilleure valeur de

nbSteps=28. De plus, en tunant encore l'hyper-paramètre **threshold** nous trouvons que le meilleur compromis entre toutes les métriques (Label-Based et Example-Based) est la valeur **0.3**.

f) Les réseaux convolutifs n'ont pas pu donner une amélioration, car nous avons atteint une limite de résultat vu la nature du problème "Multi labels avec 27 classes " et vu les données que nous possédons. Les genres comme World, Music africaine, etc… seront toujours présents pour baisser les résultats vu la pauvreté de données et la difficulté de capturer des régularités sur les données impliquant ces genres.

6. Problèmes et difficultés

Nous avons eu plusieurs problèmes relatifs à la mémoire vive , que nous allons groupé en deux points :

1. **Passage Dataframe => matrice X** : Démarrer du DataFrame représentant les traces de diffusion **(Figure 4.a.i 1)** qui consommait +4GB de RAM pour pouvoir construire le tenseur X **(illustré juste après cette figure)** était fastidieux . Sachant que la quasi totalité des fonctions proposées par les librairies Pandas, numpy … sont des fonctions qui créent une copie de leurs paramètres et qui utilisent des variables intermédiaires. Ce qui engendre un débordement mémoire .

2. **Représentation de la matrice X :** Comment représenter la matrice X ? car créer un np.zeros ((1537 , 358 , 2683)) prenait +11GB de RAM. Ce dernier était censé représenter notre Dataset Final que nous pourrons ensuite grouper en nbSteps=1, ou 2 ou … 358 pour faire notre apprentissage.

Nous avons réfléchi à plusieurs approches pour remédier à cette problématique :

- (1) Abandonner les fonctions prédéfinies des bibliothèques et coder avec des boucles et avec du code brut " from scratch ".

- (2) Jouer sur le typage des données car les array et tensors créés par défaut prenaient généralement des types de taille 64bits, alors que le type int16 pour notre matrice lui suffit largement vu les valeurs de nusers.

- (3) Utiliser les représentations Sparse => Problème de modifications de valeurs ! En général les valeurs des matrices sparses proposées par les différentes bibliothèques sont immuables, car elles possèdent pas de systèmes d'indices. Elle serve juste à stocker une matrice sparse et l'utiliser en lecture. Alors que nous avions besoin de faire tout un preprocessing nécessitant des écritures pour arriver à notre tenseur final qui peut être représenté en sparse.

- (4) transformer le DataFrame **(Figure 4.a.i 1)** en un dictionnaire. Ensuite, parcourir ce dernier en faisant des agrégations et créer notre matrice X .Plutôt que les faire sur les matrices ou dataframe denses eux même.

- (5) Dans le code , supprimer les variables qui ne sont plus utiles et lancer le "garbage collector" d'un traitement à un autre pour récupérer de la mémoire entre "code snippets".

Nous avons fait une combinaison entre (1), (2), (4) et (5) dans notre processus expérimental.

7. Conclusion

Ce projet était très enrichissant. Nous ne connaissions rien du tout sur ce qui concerne les réseaux de neurones, la classification multi-labels et ses difficultés... Grâce à ce projet nous avons largement pu initier les réseaux de neurones, maîtriser les problèmes de classification multi-labels et répondre aux problèmes de capacités de calculs et de mémoire vu la taille des données.
Nous tenons aussi à proposer des idées d'améliorations futures :

- Essayer de résoudre définitivement le problème de la mémoire afin d'explorer le nbSteps maximal=358, avec des approches de fichiers légers **pytorch,** selon les points suivants :
  - ➢ Construire le dataset final **X** de taille (1537,358,2683) sur une autre machine puissante (Plus de RAM, GPU et CPU plus puissants).
  - ➢ Le sauvegarder dans un fichier de format "pt" ou "HDF5" … supporté par Pytorch.
  - ➢ Exploiter la force des dataloader et charger en mémoire selon les besoins.

- Vu que les données sont sous forme de série temporelle, nous pouvons réfléchir à une modélisation temporelle qui correspond à des modèles un peu plus avancés et adaptés à la nature des séries temporelles comme les **RNN** avec ses variantes LSTM, GRU ...

8. Références

[1] Deep Lizard Team , "Convolutional Neural Networks (CNNs) explained." [Online].Available:http://deeplizard.com/learn/video/YRhxdVk_sIs,December 9, 2017 [Accessed: 03-Feb-2019].

[2] Ivan Vasilev , Daniel Slater , Gianmario Spacagna , Peter Roelants ,Valentino Zocca . "Neural Networks," in Python deep learning, 2nd ed., Pravin Dhandre , Ed. Birmingham : Packt Publishing Ltd., 2019, pp.34-68.

[3] Ivan Vasilev , Daniel Slater , Gianmario Spacagna , Peter Roelants ,Valentino Zocca . “Deep learning fundamentals,” in Python deep learning, 2nd ed., Pravin Dhandre , Ed. Birmingham : Packt Publishing Ltd., 2019, pp.68-93.

[4] Ivan Vasilev , Daniel Slater , Gianmario Spacagna , PeterRoelants , ValentinoZocca .“Computer vision with convolutional neural network,” in Python deep learning, 2nd ed., Pravin Dhandre , Ed. Birmingham : Packt Publishing Ltd., 2019, pp.93-122.

[5] H. Bahuleyan, “Music Genre Classification using Machine Learning Techniques,”arXiv:1804.01149 [cs, eess], Apr. 2018.

[6] J. Despois, “Finding the genre of a song with Deep Learning — A.I. Odyssey part. 1,” Hacker Noon, 21-Nov-2016. [Online]. Available: https://hackernoon.com/finding-the-genre-of-a-song-with-deep-learning-da8f59a61194. [Accessed: 21-Mar-2019].

[7] S. Kim, D. Kim, and B. Suh, “Music Genre Classification Using Multimodal Deep Learning,” in Proceedings of HCI Korea, South Korea, 2016, pp. 389–395.

[8] J.-F. Mari, F. L. Ber, and M. Benoît, “Segmentation temporelle et spatiale de données agricoles,”presented at the Actes des 6èmes Journées Cassini 2002, 2002, pp. 251–272.

[9] Tsoumakas, Grigorios, and Ioannis Katakis. "Multi-label classification: An overview." Dept. of Informatics, Aristotle University of Thessaloniki, Greece (2006).

[10] D. A. Vega-Oliveros et al., “From spatio-temporal data to chronological networks: An application to wildfire analysis,” Proceedings of the 34thACM/SIGAPP Symposium on Applied Computing - SAC ’19, pp. 675–682, 2019.

[11] G. Atluri, A. Karpatne, and V. Kumar, “Spatio-Temporal Data Mining: A Survey of Problems and Methods,” arXiv:1711.04710 [cs], Nov. 2017.