
Machine Learning pour Python

Projet : Classification de genre musical

ENSAE 2021/2022

MASTÈRE SPÉCIALISÉ - DATA SCIENCE

MICHEL CAPOT - RÉDA TAWFIKI - JEFFREY VERDIERE

Table des matières

1 Etat de l'art	1
1.1 La classification de genre musical	1
1.2 L'apprentissage automatique pour la classification de genre musical	2
2 Implémentation	3
2.1 Les données	3
2.1.1 Le dataset	3
2.1.2 Analyse exploratoire des métadonnées	3
2.2 Première stratégie	5
2.2.1 Extraction de features	5
2.2.2 Analyse exploratoire des features	6
2.2.3 Modélisation	7
2.2.4 Analyse des résultats	9
2.3 Seconde stratégie	9
2.3.1 Extraction de features	9
2.3.2 Analyse exploratoire des features	13
2.3.3 Modélisation	14
2.3.4 Analyse des résultats	18
3 Conclusion	21

1 Etat de l'art

1.1 La classification de genre musical

La classification de genre musical permet de déterminer l'appartenance d'un son à un genre musical spécifique, à l'aide de différentes techniques d'analyse et de compréhension des signaux audio.

Pour les personnes familières avec les genres en question, la reconnaissance des genres musicaux n'est pas une tâche particulièrement difficile, comme le montre l'exemple de Gjerdingen [4]. La plupart des personnes qui connaissent bien la musique occidentale sont capables d'identifier qu'une chanson donnée de Metallica est un exemple du genre 'heavy metal' et qu'une chanson donnée de Status Quo est un exemple de musique 'rock', simplement en écoutant l'audio. Il est important de noter qu'elles sont généralement capables d'effectuer ce type de classification sans même connaître la chanson ou l'artiste en question, les notes musicales entendues étant généralement suffisantes. En effet, seul un court échantillon d'audio est généralement nécessaire, quelques secondes seulement.

La question se pose de savoir s'il est possible d'écrire un algorithme capable d'effectuer des classifications de genre aussi bien que les humains, et de savoir quel type d'approche fonctionnerait le mieux.

Ghosal et Kolekar [6] expliquent qu'il existe de nombreux cas d'usage pratiques et réels de la classification et reconnaissance automatique de genres musicaux. En effet, Pandora est une plateforme de streaming et de recommandation basée aux États-Unis qui emploie des musicologues pour annoter les chansons avec des caractéristiques de genre et de rythme, dans le but de créer de meilleures playlists de recommandation. Les utilisateurs peuvent vouloir découvrir des morceaux similaires à ceux dont ils sont déjà fans, et il est logique que les fournisseurs disposent de telles métadonnées pour leurs analyses, afin de faciliter ce type de découverte. Compte tenu de la quantité considérable et sans cesse croissante de métadonnées disponibles, et de la nécessité pour les services de gérer des bases de données de chansons de plus en plus importantes, l'analyse et l'annotation manuelles de musiques ne semble pas être une solution durable à long terme. D'un point de vue technique, malgré le coût de développement et de maintenance du système, les dépenses à long terme pourraient être considérablement réduites grâce à l'utilisation de systèmes de reconnaissance automatisés. De nombreuses grandes entreprises sont bien évidemment déjà conscientes des perspectives offertes par les techniques de reconnaissance automatique de genres, mais des questions subsistent quant à la meilleure façon d'architecturer un système de reconnaissance de genres musicaux.



De nombreux autres exemples de cas concrets de l'utilisation de la classification de genres musicaux existent. En effet, on peut prendre l'exemple de Spotify qui règne aujourd'hui sur le marché du streaming musical. Elle compte actuellement des millions de chansons dans sa base de données et possède l'un des meilleurs systèmes de recommandation basé sur la classification automatique de genre musical : le service "Discover Weekly", qui permet aux utilisateurs de découvrir hebdomadairement des nouveaux morceaux proches de leurs centres d'intérêt.

En somme, les entreprises utilisent aujourd'hui la classification musicale, soit pour pouvoir faire des recommandations à leurs clients (comme Spotify), soit simplement comme un produit (par exemple, Shazam). Les techniques d'apprentissage automatique se sont avérées très efficaces pour faire ressortir certaines tendances et pour répondre aux besoins d'une clientèle toujours plus exigeante.

1.2 L'apprentissage automatique pour la classification de genre musical

Comme nous avons pu le voir, diverses techniques d'apprentissage automatique ont déjà été appliquées à la problématique de reconnaissance et classification automatique de genre musical. Cette dernière peut être faite de différentes manières. Une des principales approches consiste à traiter un ensemble de données audio, à en extraire des features, puis à utiliser un ensemble de ces features extraites pour entraîner un classifieur d'apprentissage automatique. Cependant, il existe d'autres façons de faire et d'autre données à explorer, notamment les paroles de chansons, les critiques d'albums ou les illustrations d'albums, qui se sont aussi révélées assez efficaces. La classification "multimodale" quant à elle s'est avérée prometteuse, mais à encore perfectionner. Après avoir exploré ces différentes approches et passé en revue les documents de recherches existants dans ce domaine, nous avons décidé de nous concentrer sur la stratégie de classification de genre musical par traitement audio, car elle englobe les techniques qui se sont avérées les plus efficaces.

En général, les classificateurs d'apprentissage automatique sont performants lorsqu'ils sont capables de "comprendre" ce qui fait d'un échantillon de données un membre à part entière d'une classe particulière et, dans de nombreux cas, lorsque les classes tendent à être facilement "séparables" les unes des autres (c'est-à-dire lorsque les différentes classes varient de manière significative en termes de valeurs moyennes des caractéristiques). Lorsque les humains sont capables de reconnaître le genre d'une piste audio donnée, ils le font généralement sur la base de critères liés aux aspects du son en question, comme par exemple les instruments utilisés, la sensation avec laquelle ils sont joués, la "tendance sonore globale" de la chanson, etc. Cependant, ces types de propriétés musicales émergentes peuvent être difficiles à reconnaître pour un logiciel. Néanmoins, étant donné la disponibilité directe des fichiers audio, il est possible pour les logiciels de détecter certains aspects des sons que les humains ne peuvent pas percevoir.

Notre projet met en avant l'utilisation de l'apprentissage automatique et sa capacité à classer les chansons avec précision.

Tout d'abord, les différentes méthodes d'extraction des caractéristiques des fichiers audio ont été décrites dans notre rapport. Une explication des caractéristiques extraites et une analyse des caractéristiques choisies (en termes d'utilité) sont ensuite mises en avant. Par la suite, nous avons choisi de décrire les techniques de classification et de mettre en avant le modèle d'apprentissage automatique que nous avons retenu. Enfin, nous avons procédé à l'interprétation des résultats, ainsi qu'une évaluation des choix de conception. Notre étude se terminera par les principales leçons que nous avons apprises, et ce que nous aurions pu faire pour potentiellement améliorer nos recherches.

2 Implémentation

2.1 Les données

2.1.1 Le dataset

Après de longues réflexions autour de la problématique que nous souhaitions traiter, nous avons opté pour le dataset FMA [2] , libre d'accès sur Internet. Les Archives de Musique Libre (FMA) sont un ensemble de données récoltées à grande échelle permettant la recherche d'information musicale. Le dataset entier contient plus de 343 jours d'audio provenant de 106 574 pistes, de 16 341 artistes et 14 854 albums, classés en 161 genres. Elle fournit des données audio complètes et de haute qualité, des caractéristiques précalculées, ainsi que des métadonnées concernant les pistes. Nous nous focaliserons uniquement sur la partie audio pour réaliser notre classification de genre musical.

Il existe quatre sous-ensembles de dataset définis par les auteurs du FMA :

- Le dataset Full : l'ensemble de données complet contenant 106 574 pistes non rognées et 161 genres non équilibrés (879 GiB) ;
- Le dataset Large : l'ensemble de données complet avec l'audio limité à des clips de 30 secondes extraits du milieu des pistes (ou de la piste entière si elle est plus courte que 30 secondes) (161 genres non équilibrés (93 GiB)) ;
- Le dataset Medium : une sélection de 25 000 clips de 30 secondes ayant un seul genre racine (22 GiB) ;
- Le dataset Petit : un sous-ensemble équilibré contenant 8 000 pistes de 30 secondes et 8 genres équilibrés (7.2 GiB)

La répartition officielle en ensembles de train, de validation et de test (80/10/10) utilise un échantillonnage stratifié pour préserver le pourcentage de pistes par genre. Les chansons d'un même artiste ne font partie que d'un seul ensemble.

A noter que pour que le signal puisse être entièrement reconstruit à partir des échantillons, il faut et il suffit que la fréquence d'échantillonnage soit strictement supérieure à deux fois la plus grande fréquence présente dans le spectre du signal continu. Chacun des extraits mp3 de 30 secondes de notre dataset a donc subit un échantillonnage à 44.1kHz, ce qui permet le respect du critère de Shannon puisque la plus haute fréquence audible est à 20kHz.

Compte tenu du volume très important des données et au vue des machines que nous avons à notre disposition, il aurait été irréaliste de travailler avec autre chose que l'échantillon "Petit".

2.1.2 Analyse exploratoire des métadonnées

Comme nous pouvons le constater sur ce diagramme camembert, l'équilibre entre les classes dans notre dataset est quasiment parfait, avec près de 1000 morceaux pour chacune d'entre elles :

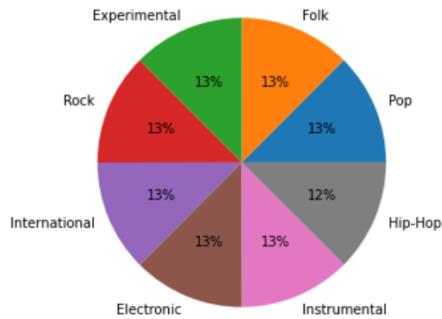


FIGURE 1 – Répartition des classes dans le DataSet

Nous questionnons cependant l'homogénéité desdites classes, dans la mesure où les morceaux de ce dataset appartiennent initialement à des catégories plus fines de sous-genres. Plusieurs valeurs de sous-genres peuvent être rattachées à un même morceau, comme le montre le diagramme en barres suivant :

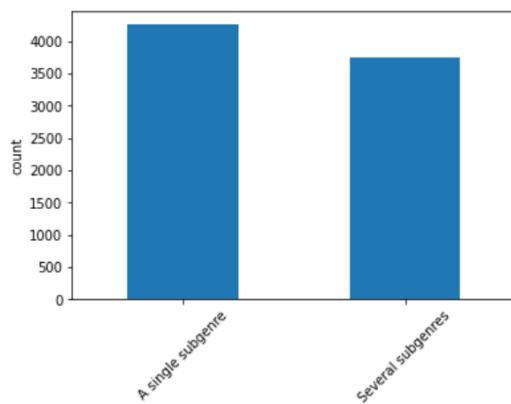


FIGURE 2 – Décompte des morceaux associés à un sous-genre unique *vs* plusieurs sous-genres

Une partie non-négligeable des données (près de la moitié) se prêterait donc à une classification multilabel, mais le nombre de sous-genres (161 au total) rend cette tâche impossible compte tenu de la faible taille de notre échantillon. En l'absence de stratégie rigoureuse pour ne retenir qu'un seul sous-genre par morceau, nous allons donc nous focaliser sur les genres de plus haut niveau (les huit genres présent sur le diagramme camembert ci-dessus), bien que nous soyons maintenant conscients de l'hétérogénéité de ces classes, ou du moins de certaines d'entre elles, comme le confirme le diagramme en barres suivant, qui représente le nombre de valeurs différentes de sous-genre au sein de chaque catégorie :

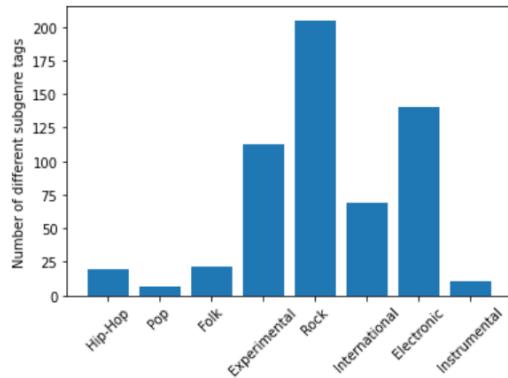


FIGURE 3 – Nombre de combinaisons différentes de sous-genres par grande catégorie musicale

La représentation en arborescence des sous-genres, avec le nombre de morceaux du dataset qui leur sont associés (disponible dans le notebook dans la section Exploratory Metadata Analysis), souligne également cette hétérogénéité : au sein de nos huit catégories principales, pratiquement tous les sous-genres (y compris sur les niveaux plus profonds, le cas échéant) sont représentés dans notre dataset, ce qui présage une généralisation difficile de l'apprentissage pour les modèles que nous allons construire.

2.2 Première stratégie

2.2.1 Extraction de features

Gartzman [3] explique que pour représenter un son, on trace généralement, l'évolution de son amplitude en fonction du temps. On utilise souvent la transformation de Fourier qui permet de faire apparaître l'amplitude en fonction de la fréquence. Cependant, des études ont montré que les humains ne perçoivent pas les fréquences sur une échelle linéaire. Nous détectons mieux les différences dans les basses fréquences que les hautes fréquences. Par exemple, nous savons faire la différence entre 500 et 1000 Hz mais on ne sait pas vraiment faire la différence entre 10 000 et 10 500 Hz. Néanmoins, l'oreille humaine se concentre que sur un très faible spectre d'onde sonore. Pour se concentrer sur des spectres qui correspondent à la sensation psychologique de hauteur d'un son, on utilise [9] après la transformée de Fourier, la formule analytique suivante :

$$mel(f) = 1000 \log_2 \left(1 + \frac{1}{\frac{f}{1000}} \right) \quad (1)$$

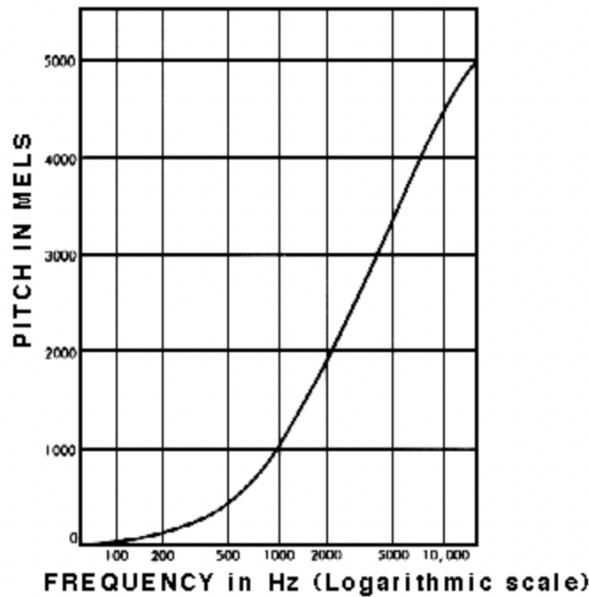


FIGURE 4 – Correspondance entre échelles Mel et Hz

L'intérêt donc de cette transformation est d'entraîner des algorithmes de machine learning dans les mêmes conditions que l'oreille humaine. On réalise alors les étapes successives suivantes :

- Extraits des sons de 10 secondes avec un décalage de 10s pour que l'extrait soit pris au "milieu" des 30s avec la bibliothèque python "Librosa" ;
- Cartographie du signal audio du domaine temporel au domaine fréquentiel à l'aide de la transformée de Fourier rapide ;
- Conversion de l'axe y en échelle logarithmique et l'amplitude sous forme de couleur en décibel pour former le spectrogramme ;
- Cartographie de l'axe y (fréquences) sur l'échelle mel pour former le spectrogramme mel et donc pour entraîner des structures dans les mêmes conditions que l'homme.

2.2.2 Analyse exploratoire des features

Nous avons implémenté une fonction "plot_melspectrograms" permettant d'afficher une série de melspectrograms pour chaque genre afin de saisir visuellement leurs caractéristiques distinctives ; en fin de compte, ceci nous a plutôt fait remarquer leurs différences internes, comme l'illustrent les exemples suivants (pour les catégories "pop" et "experimental").

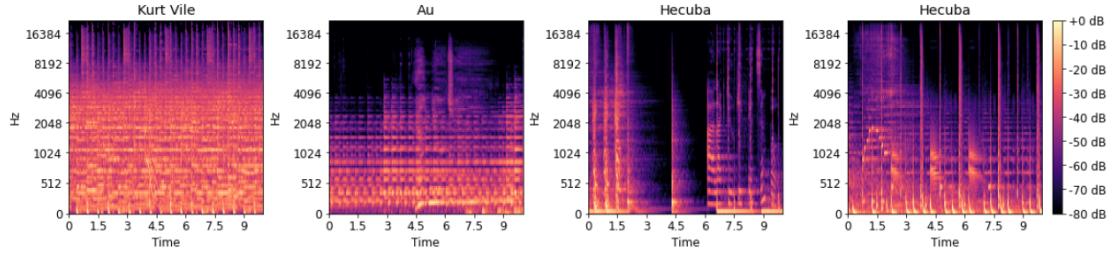


FIGURE 5 – Spectrogramme Mel pour la catégorie 'pop'

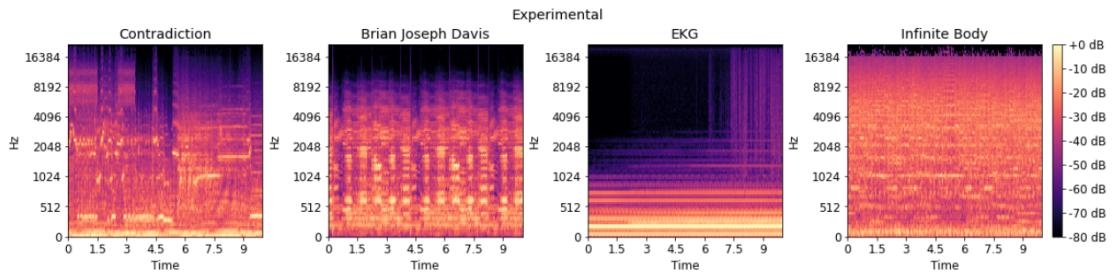


FIGURE 6 – Spectrogramme Mel pour la catégorie 'experimental'

Quoiqu'il ne s'agisse ici que de quatre morceaux par genre (ceux-ci ne sont donc pas nécessairement représentatifs), nous sommes placés une fois de plus face à un manque de cohérence intrinsèque des catégories.

2.2.3 Modélisation

Au cours des dernières années [11] explique que les meilleurs résultats en vision par ordinateur ont été obtenus avec l'architecture ResNet. En effet, les réseaux de neurones étant très difficile à entraîner, les couches cachées du réseau du RestNet50 sont plus profondes que celles utilisées dans un simple réseau convolutif. La question de l'explosion en temps de calcul lors de la descente de gradient à mener à d'autres structures.

Ce problème a été résolu en introduisant des "résidus profonds". Pour cela, ils introduisent des connexions de raccourci qui effectuent simplement des mapages comme sur la figure (7) suivante :

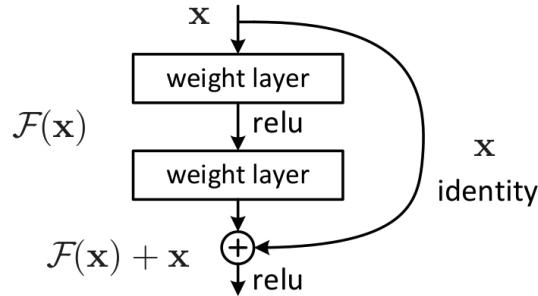


FIGURE 7 – Calcul des résidus profonds

Les couches s'adaptent à un mappage résiduel désigné par $H(x)$ et les autres couches non-linéaires s'adaptent à un autre mappage $F(x) := H(x) - x$. Le mappage d'origine devient $H(x) := F(x) + x$ comme sur la figure 7. L'avantage est qu'il n'y a pas de paramètres supplémentaires ajoutés au modèle et que le temps de calcul n'augmente pas également.

La librairie Keras développée par F.Chollet permet de mettre très facilement en place une architecture ResNet50 comme celle de la figure (8) suivante :

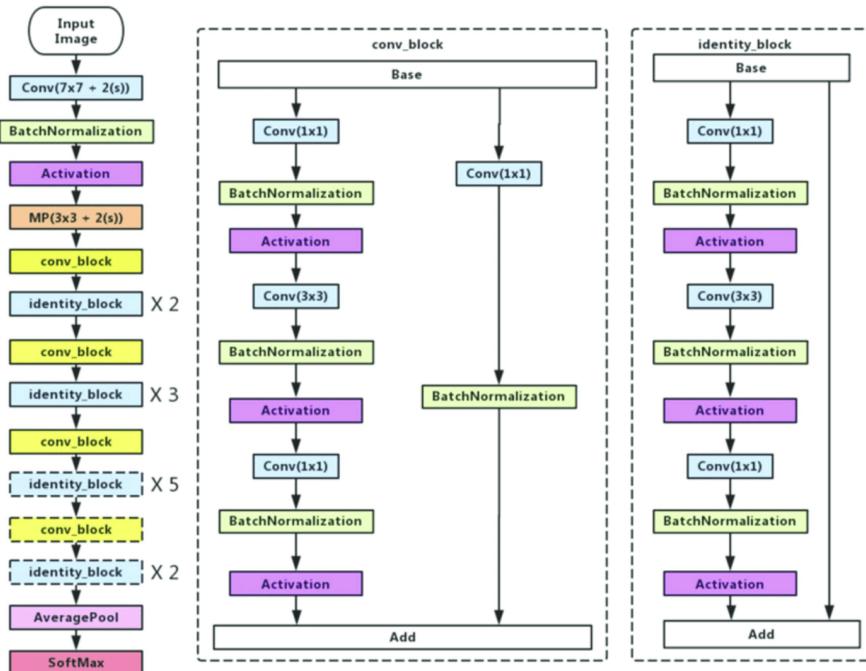


FIGURE 8 – Architecture du réseau RestNet50

L'entraînement est réalisé avec Colab sur leur propre GPU pour réduire les temps de calcul. La cross-validation serait souhaitable pour comparer différentes architectures, compte tenu du faible nombre d'individus dans le dataset, mais irréaliste au vu du temps d'entraînement d'un seul modèle. Nous avons donc fait

le choix de ne pas en faire.

2.2.4 Analyse des résultats

Avant de procéder à l'entraînement du modèle, nous avons normalisé les valeurs constitutives des spectrogrammes en les divisant par le maximum de leur valeur absolue (80), suivant ainsi la pratique usuelle en vision par ordinateur (où l'on divise, pour chaque pixel des images, la valeur de chacun des canaux par 255). Nous avons séparé notre jeu de données en échantillons d'apprentissage et de test avec stratification afin de préserver l'équilibre des classes. Nous avons opté pour la métrique d'accuracy afin d'évaluer les performances du modèle : celle-ci est la plus simple à interpréter (ratio de prédictions correctes) et ne risque pas de nous induire en erreur dans le cas présent puisque les classes sont parfaitement équilibrées.

L'entraînement du ResNet50 étant très long avec des données aussi volumineuses, nous l'avons prématièrement stoppé au vu de ses performances décevantes, que résume la capture d'écran suivante :

```
▶ resnet_history = resnet.fit(X_train, y_train, epochs=50, validation_data = (X_test, y_test), batch_size=16, shuffle=True)

Epoch 1/50
375/375 [=====] - 4651s 12s/step - loss: 2.1160 - accuracy: 0.2703 - val_loss: 2.5159 - val_accuracy: 0.1650
Epoch 2/50
375/375 [=====] - 4560s 12s/step - loss: 1.8020 - accuracy: 0.3367 - val_loss: 96.4272 - val_accuracy: 0.1655
Epoch 3/50
375/375 [=====] - 4393s 12s/step - loss: 1.6592 - accuracy: 0.3985 - val_loss: 3.5447 - val_accuracy: 0.1710
Epoch 4/50
375/375 [=====] - 4615s 12s/step - loss: 1.5983 - accuracy: 0.4232 - val_loss: 2.7295 - val_accuracy: 0.2165
Epoch 5/50
375/375 [=====] - 4526s 12s/step - loss: 1.6356 - accuracy: 0.4104 - val_loss: 1.8463 - val_accuracy: 0.4000
Epoch 6/50
375/375 [=====] - 4351s 12s/step - loss: 1.5239 - accuracy: 0.4632 - val_loss: 4.6358 - val_accuracy: 0.2275
Epoch 7/50
375/375 [=====] - 4436s 12s/step - loss: 1.4933 - accuracy: 0.4654 - val_loss: 2.3705 - val_accuracy: 0.2660
Epoch 8/50
375/375 [=====] - 4408s 12s/step - loss: 1.4591 - accuracy: 0.4821 - val_loss: 5.2685 - val_accuracy: 0.1835
Epoch 9/50
375/375 [=====] - 4454s 12s/step - loss: 1.4250 - accuracy: 0.4959 - val_loss: 4.0597 - val_accuracy: 0.1820
Epoch 10/50
375/375 [=====] - 4331s 12s/step - loss: 1.3982 - accuracy: 0.5089 - val_loss: 1.6794 - val_accuracy: 0.4015
Epoch 11/50
375/375 [=====] - 4480s 12s/step - loss: 1.3872 - accuracy: 0.5119 - val_loss: 1.7573 - val_accuracy: 0.4415
Epoch 12/50
375/375 [=====] - 4653s 12s/step - loss: 1.3561 - accuracy: 0.5254 - val_loss: 4.2734 - val_accuracy: 0.1270
Epoch 13/50
375/375 [=====] - 4605s 12s/step - loss: 1.3211 - accuracy: 0.5384 - val_loss: 12.2279 - val_accuracy: 0.1780
Epoch 14/50
82/375 [====>.....] - ETA: 55:50 - loss: 1.2595 - accuracy: 0.5640
```

FIGURE 9 – Entraînement de la structure ResNet

Au fil des epochs, nous constatons que les performances sur l'échantillon d'apprentissage s'améliorent lentement, tandis que celles qui sont réalisées sur l'échantillon de test sont de qualité très fluctuante (mais toujours en-dessous des premières). Un entraînement plus long ferait probablement ressortir une situation d'overfitting : en dépit de son architecture sophistiquée, idéale pour ce type de tâche selon l'état de l'art, notre réseau ne parvient pas à généraliser son apprentissage, ses résultats s'approchant de ceux qu'obtiendrait un classifieur naïf (c'est-à-dire un classifieur qui prédirait au hasard parmi les huit genres, donc avec un taux de $1/8 = 12.5\%$ de bonnes prédictions).

2.3 Seconde stratégie

2.3.1 Extraction de features

Nous avons ensuite opté pour une seconde stratégie mettant en avant l'utilisation d'autres techniques d'extraction de features, détaillées ci-dessous.

MFCC - Mel-Frequency Cepstrals Coefficients

Le principe de calcul des MFCC (Mel-scaled Frequency Cepstral Coefficients) est issu des recherches psychoacoustiques [5] sur la tonie et la perception des différentes bandes de fréquences par l'oreille humaine. La FFT (Transformée de Fourier) passe dans un banc de filtres à l'échelle de Mel. Cette échelle, non linéaire, tient principalement compte du fait que la perception des intervalles change suivant la zone du spectre à laquelle les hauteurs qui les composent appartiennent. Le principal intérêt de ces coefficients est d'extraire des informations pertinentes en nombre limité en s'appuyant à la fois sur la production (théorie Cepstale) et à la fois sur la perception de la parole (échelle des Mels).

Le calcul se déroule de la manière suivante :

- La FFT est calculée sur le fragment (frame) ;
- Cette dernière est filtrée par un banc de filtres triangulaires répartis le long de l'échelle de Mel ;
- Le logarithme module de l'énergie de sortie du banc de filtres est calculé ;
- Une Transformée en Cosinus Discrète inverse, (équivalente à la FFT inverse pour un signal réel) est appliquée ;
- Seul les premiers coefficients sont conservés.

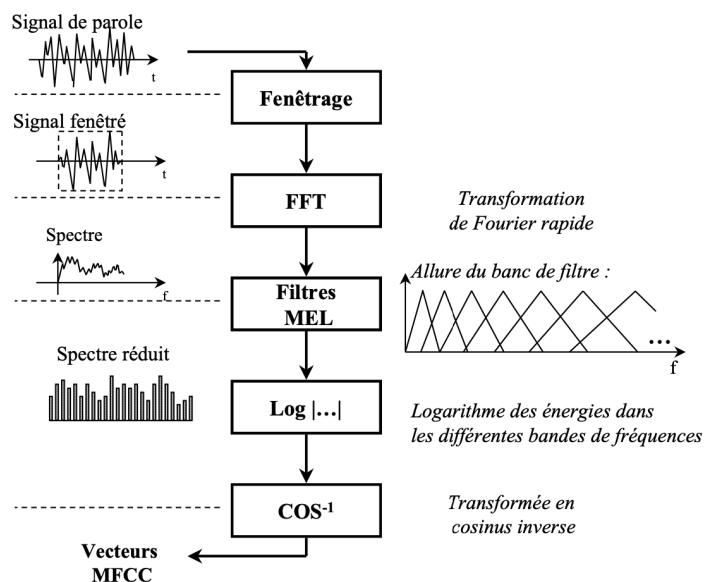


FIGURE 10 – Méthodologie de calcul des MFCC

Chromas STFT - Short-Time Fourier Transformation

Un chromagramme représente l'intensité de chaque note présente dans le signal audio à un instant donné [1]. Il est constitué de vecteurs de chroma qui sont des vecteurs à 12 dimensions représentant les 12 demi-tons de la gamme chromatique. Le chromagramme ne tient pas compte de la position des notes par rapport aux différentes octaves. Les hauteurs séparées par un nombre entier d'octaves sont perçues comme "sonnante" de manière équivalente et ont le même nom. On dit qu'elles partagent le même chroma. L'ensemble des notes partageant un même chroma est appelé classe de hauteur ou pitch class.

Dans l'espace des classes de hauteur, il n'y a pas de distinction entre les notes qui sont séparées par un nombre entier d'octaves. Par exemple LA3 (440 Hz), LA4 (880 Hz) et LA5 (1760 Hz) appartiennent à la même classe de hauteur.

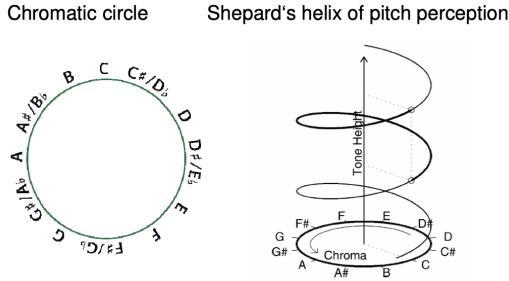


FIGURE 11 – Hélice de Shepard pour la perception de la hauteur

La figure suivante montre l'exemple du chromatogramme des quelques premières secondes du morceau Misery des Beatles. On peut voir que les notes DO, FA et LA sont présentes dans le signal.

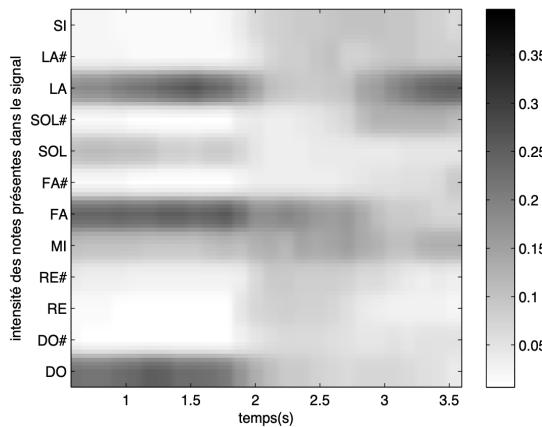


FIGURE 12 – Chromagramme d'un morceau des Beatles

La STFT représente des informations sur la classification de la hauteur et de la structure du signal. Le Chroma STFT utilise donc la transformation de Fourier à court terme pour calculer les Chroma features. A noter que pour un demi-ton et un intervalle de temps donnés, le Chroma STFT correspond à la moyenne de l'énergie spectrale correspondant à toutes les fréquences associées à ce demi-ton.

Spectral centroids

Le centroïde spectral est un descripteur acoustique dont la valeur est obtenue à partir d'un calcul statistique effectué sur la représentation fréquentielle du signal [7]. Le centroïde est donc un descripteur spectral et correspond à un type précis d'évaluation de la courbe (ou forme) du spectre.

La mesure du centroïde consiste à déterminer l'endroit où se situe le centre de gravité spectral du signal. Sa valeur correspond à une fréquence (en Hz), de part et d'autre de laquelle l'énergie du spectre est équitablement répartie. Habituellement, on calcule successivement la fréquence correspondant au centroïde sur plusieurs fenêtres temporelles (d'une durée pré-déterminée d'environ 50-60 millisecondes, en moyenne), puis on calcule la fréquence moyenne pour l'ensemble de la durée du son évalué.

Autrement dit, le centroïde "représente le poids relatif des fréquences aiguës et graves". Plusieurs études en

psychoacoustique ont démontré qu'il était fortement corrélé à la brillance perçue du son, et qu'il s'agissait d'une des propriétés les plus importantes lorsque l'oreille humaine doit différentier plusieurs timbres instrumentaux (par exemple, le timbre du hautbois et celui du cor).

Par ailleurs, on observe souvent une variation du centroïde lorsqu'un.e instrumentiste augmente la nuance dynamique de son instrument. Si un.e trompettiste joue une note d'abord en nuance piano, puis en nuance forte, l'intensité sonore (en décibels) augmentera, mais on observera également un déplacement du centroïde vers les aigus, ce qui correspond à un son plus brillant. Dans le cas d'instruments dont le registre dynamique est restreint, comme la flûte traversière, on utilise d'ailleurs souvent les variations de brillance (donc de centroïde) afin de simuler de plus grands changements dynamiques.

D'un point de vue mathématique, le centroïde spectral et l'étalement spectral sont deux mesures simples de la position et de la forme du spectre. Le centroïde spectral est le centre de "gravité" du spectre. La valeur du centroïde spectral, C_i , de la i-ème trame audio est définie comme suit :

$$C_i = \frac{\sum_{k=1}^{Wf_L} kX_i(k)}{\sum_{k=1}^{Wf_L} X_i(k)}.$$

L'étalement spectral est le deuxième moment central du spectre. Pour le calculer, il faut prendre la déviation du spectre par rapport au centroïde spectral, selon l'équation suivante :

$$S_i = \sqrt{\frac{\sum_{k=1}^{Wf_L} (k-C_i)^2 X_i(k)}{\sum_{k=1}^{Wf_L} X_i(k)}}.$$

Spectral roll-off

Le point de roll-off (traduction du terme anglais spectral roll-off point) est la fréquence au-dessous de laquelle 95% ou 85% de l'énergie spectrale est contenue [10]. Considéré comme un indice de répartition du spectre, le point de roll-off est plus grand pour les signaux ayant un spectre important au niveau des hautes fréquences. Généralement, il permet de distinguer les signaux bruités des signaux harmoniques. Nous utiliserons la définition suivante :

$$\sum_0^{ROF} a^2(k) = 0.85 \sum_0^{F_e} a^2(k)$$

où ROF et F_e sont respectivement le point de roll-off et la fréquence d'échantillonage.

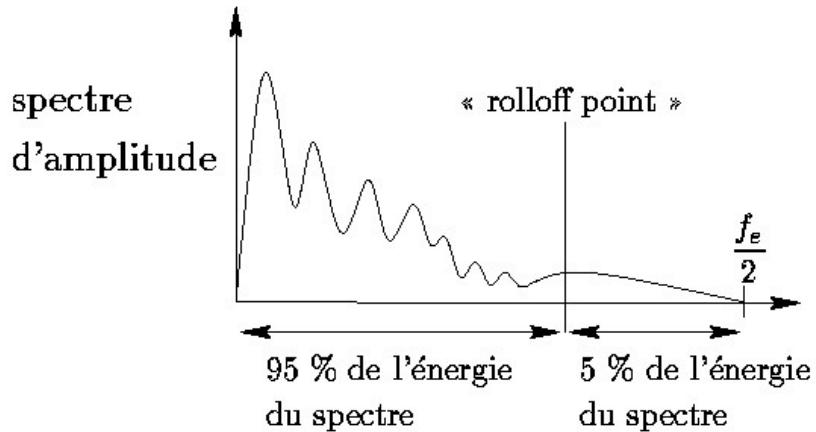


FIGURE 13 – Point de Roll-Off à 95% sur le spectre d'amplitude

NB : Le principe est le même à 85%.

Zero Crossing Rate

Le ZCR (Zero Crossing Rate) [8] ou « Taux de passage à zéro » permet simplement d'évaluer le nombre de fois qu'un signal passe par zéro (ou change de signe) dans un certain intervalle de temps, ce qui permet d'estimer la fréquence instantanée du signal.

On pose $\text{signe}(x) = 1$ si $x \geq 0$ et $\text{signe}(x) = -1$ si $x < 0$. Le zero crossing rate est la fonction ZCR de $\mathbb{N} \rightarrow \mathbb{R}$ telle que :

$$zcr(n) = \sum_{m=-\infty}^{+\infty} \frac{1}{2} \times |\text{signe}(sf(n, m)) - \text{signe}(sf(n, m - 1))|$$

Comme précédemment,

$$zcr(n) = \sum_{m=n-N/2}^{n+N/2} \frac{1}{2} \times |\text{signe}(sf(n, m)) - \text{signe}(sf(n, m - 1))|$$

2.3.2 Analyse exploratoire des features

Après avoir extrait ces features, nous avons souhaité évaluer à nouveau l'homogénéité de nos classes en comparant leurs distributions respectives. En voici deux exemples, réalisés avec des diagrammes en violons sur la moyenne du premier chroma (la note SI) et la moyenne du spectral roll-off :

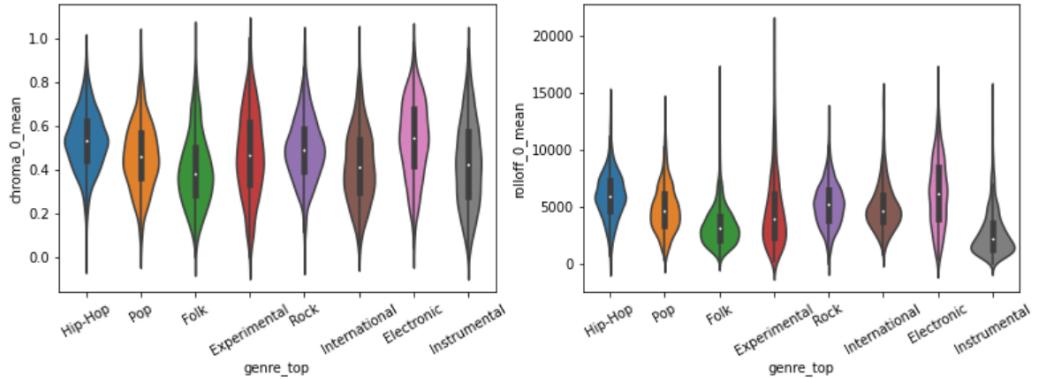


FIGURE 14 – Diagrammes en violon réalisés sur le premier chroma et la moyenne spectrale du roll-off

Sur le premier graphique, nous constatons que les huit distributions sont très aplatis, indiquant une forte variance interne ; les écarts interquartiles ne semblent pas sensiblement différents d'un genre musical à l'autre, ce qui annonce à nouveau des difficultés pour les distinguer. Le second graphique, lui, montre des distributions plus condensées, avec des "bosses" situées à des fréquences différentes d'un genre à l'autre; en revanche, la queue allongée des violons indique, pour tous les genres, la présence d'outliers susceptibles de biaiser l'apprentissage de nos modèles.

2.3.3 Modélisation

kNN

Le k-NN (k-Nearest Neighbours) est un algorithme standard de classification qui repose exclusivement sur le choix de la métrique de classification. Il est « non paramétrique » (seul k doit être fixé) et se base uniquement sur les données d'entraînement.

L'idée est la suivante : à partir d'une base de données étiquetées, on peut estimer la classe d'une nouvelle donnée en regardant quelle est la classe majoritaire des k données voisines les plus proches (d'où le nom de l'algorithme). Le nombre de voisins k à considérer est donc le seul paramètre à fixer.

Les métriques les plus souvent choisies sont la distance usuelle dite euclidienne (comme dans la figure) et la distance de Mahalanobis (qui tient compte de la variance – du point de vue statistique – et de la corrélation entre les données). Bien que l'algorithme puisse fonctionner avec ces métriques par défaut, il est généralement bien meilleur quand il est utilisé avec une métrique adaptée aux données, métrique qui peut être calculée à partir d'heuristiques connues liées au problème (par exemple la distance euclidienne pondérée).

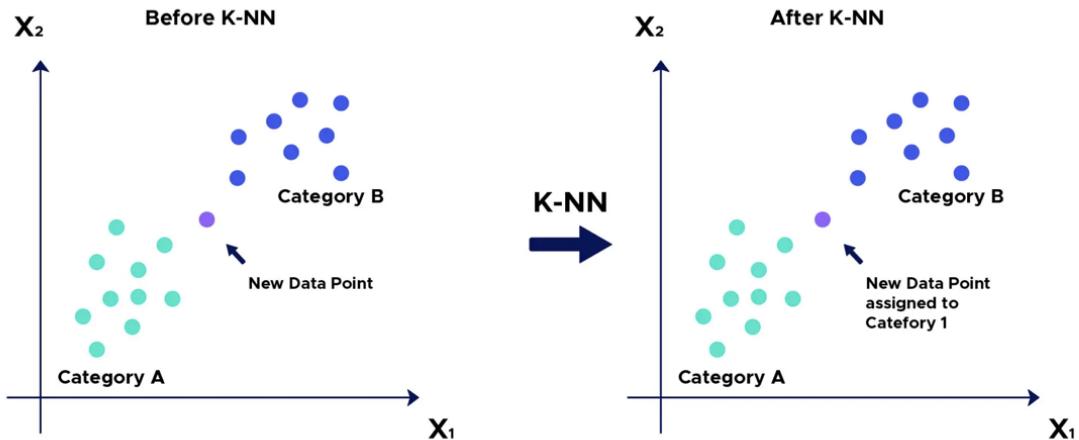


FIGURE 15 – Description graphique de l'algorithme KNN

En résumé, les grandes étapes de l'algorithme sont les suivantes :

- Etape 1 : sélectionner le nombre de k-voisins ;
- Etape 2 : calculer la distance ;
- Etape 3 : prendre les k-voisins les plus proches selon la distance calculée ;
- Etape 4 : parmi ces k-voisins, comptez le nombre de points appartenant à chaque catégorie ;
- Etape 5 : attribuer le nouveau point à la catégorie la plus présente parmi ces k-voisins.

SVM

Les machines à vecteurs de support, ou support vector machine (SVM), sont des modèles de machine learning supervisés centrés sur la résolution de problèmes de discrimination et de régression mathématiques. Elles ont été conceptualisées dans les années 1990 à partir d'une théorie d'apprentissage statistique développée par les informaticiens russes Vladimir Vapnik et Alexey Chervonenkis : la théorie de Vapnik-Chervonenkis. Ce modèle a été rapidement adopté en raison de sa capacité à travailler avec des données de grandes dimensions, ses garanties théoriques et les bons résultats réalisés en pratique. Requérant un faible nombre de paramètres, les SVM sont appréciées pour leur simplicité d'usage.

Le principe des SVM consiste à ramener un problème de classification ou de discrimination à un hyperplan (feature space) dans lequel les données sont séparées en plusieurs classes dont la frontière est la plus éloignée possible des points de données (ou "marge maximale"). D'où l'autre nom attribué aux SVM : les séparateurs à vaste marge.

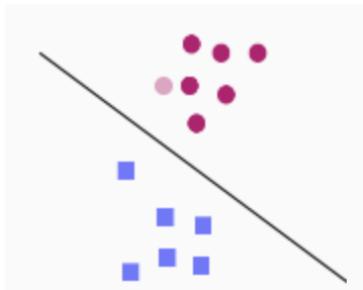


FIGURE 16 – Description graphique du SVM

Le concept de frontière implique que les données soient linéairement séparables. Pour y parvenir, les support vector machines font appel à des noyaux, c'est-à-dire des fonctions mathématiques permettant de projeter et séparer les données dans l'espace vectoriel, les "vecteurs de support" étant les données les plus proches de la frontière. C'est la frontière la plus éloignée de tous les points d'entraînement qui est optimale, et qui présente donc la meilleure capacité de généralisation.

RandomForest

L'algorithme des 'forêts aléatoires' (ou Random Forest) est un algorithme de classification qui réduit la variance des prévisions d'un arbre de décision seul, améliorant ainsi leurs performances. Pour cela, il combine de nombreux arbres de décisions dans une approche de type bagging.

En effet, l'algorithme des 'forêts aléatoires' effectue un apprentissage en parallèle sur de multiples arbres de décision construits aléatoirement et entraînés sur des sous-ensembles de données différents. Le nombre idéal d'arbres, qui peut aller jusqu'à plusieurs centaines voire plus, est un paramètre important : il est très variable et dépend du problème. Concrètement, chaque arbre de la forêt aléatoire est entraîné sur un sous ensemble aléatoire de données selon le principe du bagging, avec un sous ensemble aléatoire de features (caractéristiques variables des données) selon le principe des « projections aléatoires ». Les prédictions sont ensuite moyennées lorsque les données sont quantitatives ou utilisés pour un vote à la majorité pour des données qualitatives, dans le cas des arbres de classification, ce qui est notre cas ici. L'algorithme des forêts aléatoires est connu pour être un des classificateurs les plus efficaces « out-of-the-box » (c'est-à-dire nécessitant peu de prétraitement des données).

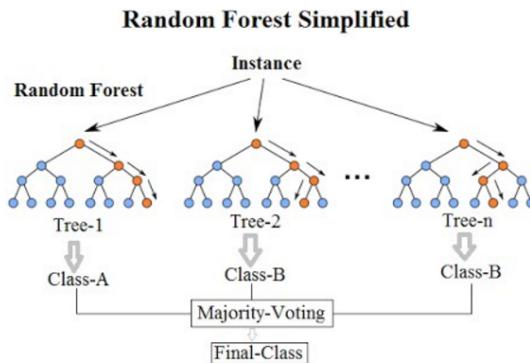


FIGURE 17 – Description graphique du Random Forest

Perceptron Multicouches

Un perceptron multicouches, appelé Multi-Layered Perceptron, noté MLP, est constitué d'unités de calcul élémentaires appelés neurones. Un neurone possède des entrées, qui sont des variables à valeur réelles, notées x_1, \dots, x_n , et une sortie, notée y . Chaque entrée est associée à un poids, noté w_i . Le calcul effectué par un neurone consiste à multiplier la valeur de chaque entrée x_i par son poids w_i et d'en faire la somme. Le résultat de cette somme est additionné à une constante b (appelé le biais). Ce résultat constitue lui-même l'entrée d'une fonction non linéaire f , appelée fonction d'activation. Le résultat de ce calcul constitue la sortie \hat{y} du neurone :

$$\hat{y} = f(\sum_{i=1}^n w_i x_i + b)$$

Graphiquement, le perceptron multicouches peut être représenté comme suit :

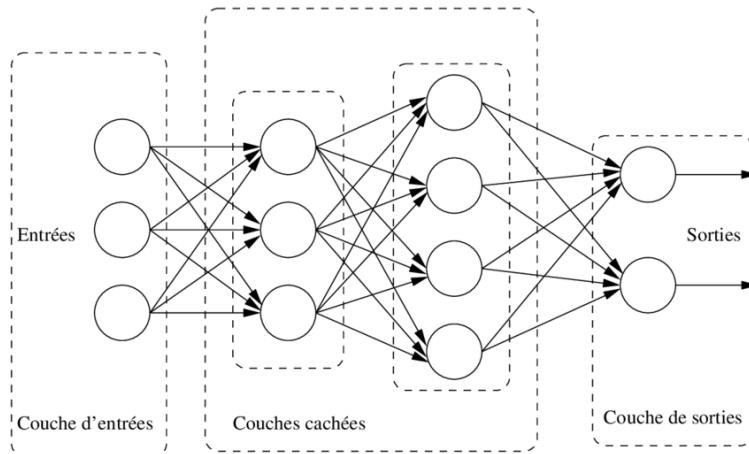


FIGURE 18 – Structure usuelle d'un perceptron multicouche

GridSearchCV

La plupart des modèles de Machine Learning doivent être paramétrés pour donner les meilleurs résultats. Par exemple pour un Random Forest, on doit choisir le nombre d'arbres à créer et le nombre de variables à utiliser à chaque division d'un noeud. Si on paramètre à la main, cela peut vite s'avérer très coûteux en temps (et pas forcément très intéressant).

C'est là que le Grid search intervient. C'est une méthode d'optimisation (hyperparameter optimization) qui va nous permettre de tester une série de paramètres et de comparer les performances pour en déduire le meilleur paramétrage.

Il existe plusieurs manières de tester les paramètres d'un modèle et le Grid Search est une des méthodes les plus simples. Pour chaque paramètre, on détermine un ensemble de valeurs que l'on souhaite tester. Par exemple, dans le cas d'un Random Forest on pourrait tester :

- Nombre d'arbres : 100, 150, 200, 250 ;
- Nombre de variables sélectionnées : 8, 10, 12, 14, 16.

Le Grid Search croise simplement chacune de ces hypothèses et va créer un modèle pour chaque combinaison de paramètres (ici, 20).

Il faut maintenant tester ces 20 modèles sur le même dataset. Une méthode de validation croisée est alors adaptée à la situation : le k-fold. La méthode consiste à découper le data set en k échantillons. On sélectionne x échantillons pour constituer l'échantillon d'apprentissage. Les k-x échantillons restants permettront d'évaluer la performance du modèle. Pour construire le modèle suivant on sélectionne les échantillons différemment de manière à ne jamais avoir les mêmes échantillons d'apprentissage et de validation.



FIGURE 19 – Méthode de Validation Croisée utilisée dans un GridSearch

2.3.4 Analyse des résultats

Avant de procéder à l’entraînement du modèle, nous avons normalisé les valeurs constitutives des spectrogrammes au moyen d’un RobustScaler, afin de limiter la sensibilité de l’apprentissage aux outliers que nous avons détectés. Nous avons une nouvelle fois séparé notre jeu de données en échantillons d’apprentissage et de test avec stratification afin de préserver l’équilibre des classes, et opté à nouveau pour la métrique d’accuracy pour les mêmes raisons.

Tous les modèles entraînés dans le GridSearchCV ont des performances relativement stables d’un échantillon d’apprentissage à l’autre ; si leurs résultats sont, en moyenne, meilleurs que ceux du ResNet50 utilisé dans la première stratégie, ils restent assez médiocres, avec un score d’accuracy dépassant à peine 45% pour le meilleur classifieur (un Random Forest construit sur 500 arbres de décision de profondeur maximale 50 avec au plus 10 features examinés à chaque nœud). Voici les features contribuant le plus (en moyenne) à la réduction de l’impureté des nœuds dans chaque arbre, par ordre décroissant d’importance :

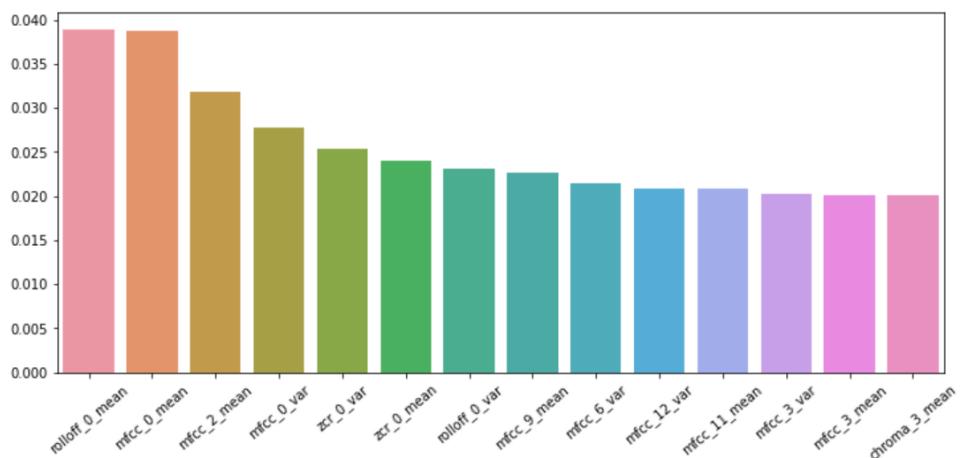


FIGURE 20 – Random Forest Feature Importance

Le perceptron multicouche entraîné par la suite (avec une couche de Dropout insérée entre les deuxième et troisième couche cachées pour prévenir l'overfitting) ne permet pas d'obtenir de meilleurs résultats ; l'entraînement est arrêté prématurément au terme de 31 epochs puisque l'accuracy sur l'échantillon de test cesse de décroître. Le graphique suivant récapitule la progression de l'apprentissage et les performances du réseau sur les deux échantillons, illustrant une tendance à l'overfitting dès la 10^e epoch :

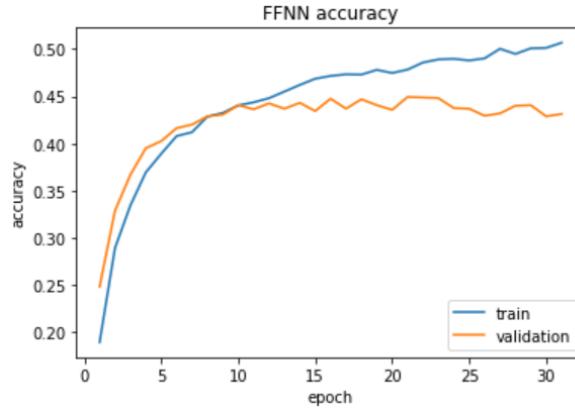


FIGURE 21 – Graphique de performances du réseau de neurones

Ayant préalablement mis en évidence l'homogénéité variable des huit genres musicaux, nous avons souhaité étudier les performances des modèles au cas par cas, en effectuant pour chacun d'eux un calcul spécifique d'accuracy dont nous rendons compte dans les deux diagrammes en barres suivants :

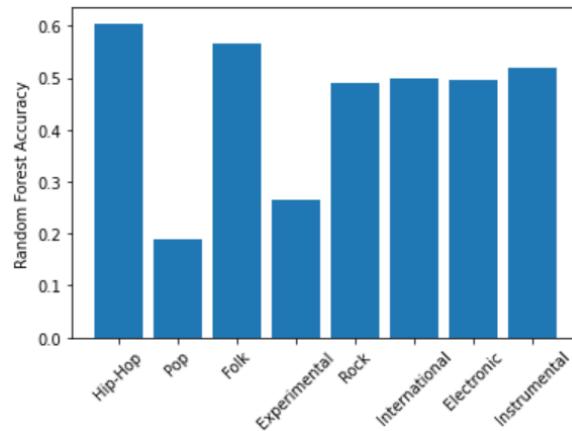


FIGURE 22 – Performances de Random Forest sur les différents sous-genres

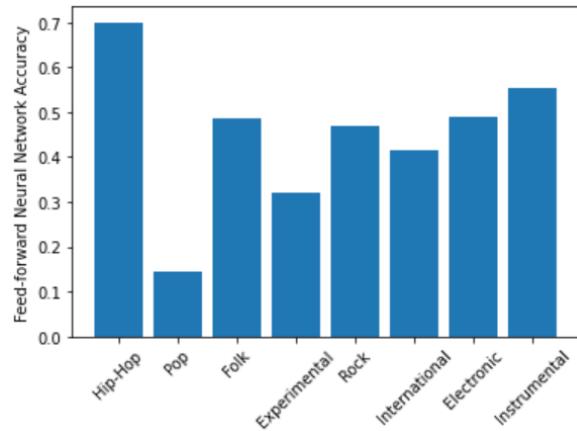


FIGURE 23 – Performances perceptron multicouche sur les différents sous-genres

Nous constatons comme attendu que les performances sont très variables d'un genre musical à l'autre : le hip-hop, genre relativement homogène dans le dataset, obtient des résultats décents (respectivement près de 60 et 70% d'accuracy pour le Random Forest et le perceptron multicouche), tandis que la pop et l'experimental, qui sont des catégories plus « fourre-tout », mettent nos modélisations en échec. Le score d'accuracy calculé à l'étape précédente sur tout l'échantillon de test ne reflétait donc qu'une tendance moyenne. Notons que les performances du réseau de neurones semblent meilleures que celles du Random Forest pour les genres plus homogènes, et moins bonnes sur les genres les plus hétérogènes.

3 Conclusion

Malgré un suivi rigoureux des méthodes canoniques en Music Information Retrieval et la mise en œuvre d'une démarche logique, les résultats que nous obtenons s'avèrent décevants. Même si nos différents classifiants font mieux qu'une prédiction aléatoire, puisqu'ils déterminent le bon label près d'une fois sur deux, un tel ratio est très éloigné des standards de la discipline rapportés par Won et al.[12].

Nous attribuons ces mauvaises performances à la qualité et au nombre insuffisants des données que nous avons utilisées pour entraîner nos différents modèles : à cette échelle, les catégories de genre top-level semblent trop hétérogènes pour permettre une classification plus efficace (à l'exception du hip-hop), et nos ressources computationnelles limitées ne nous ont pas permis de tirer avantage de la taille substantielle du dataset FMA.

Malgré ces résultats insatisfaisants, ce projet de Machine Learning en Python a tout de même été une expérience enrichissante : grâce à la prise en main des bibliothèques dédiées, nous avons pu nous familiariser avec les tâches d'audio processing (et plus spécifiquement la classification de genre musical) en appliquant plusieurs méthodes d'extraction de features. De surcroît, les difficultés que nous avons rencontrées nous ont incités à élaborer une stratégie plus fine et réflexive, et nous ont sans doute permis de cette manière de nous rapprocher davantage de la réalité du métier de data scientist.

Références

- [1] Müler BALKE. “Short-Time Fourier Transform and Chroma Features”. In : (2020).
- [2] Defferrard Benzi Vandergheynst BRESSION. “FMA : A Data Set for Music Analysis”. In : (2017).
- [3] GARTZMAN. “Getting to Know Mel Spectrogram”. In : (2019).
- [4] GJERDINGEN. “Scanning the Dial : The Rapid Recognition of Music Genres”. In : (2008).
- [5] Majeed Husain Samad IDBEA. “Mel frequency cepstral coefficients (Mfcc) feature extraction enhancement in the application of speech recognition : A comparison study”. In : (2015).
- [6] Ghosal KOLEKAR. “Music Genre Recognition Using Deep Neural Networks and Transfer Learning”. In : (2018).
- [7] Kua Husain Thiruvaran NOSRATI. “Investigation of Spectral Centroid Magnitude and Frequency for Speaker Recognition”. In : (2010).
- [8] Gouyon PACHET. “On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds”. In : (2002).
- [9] RICHARD. “Reconnaissance et interaction vocale”. In : (2019).
- [10] Yu You Yang Liu Li Yang SHAN. “Spectral Roll-off Points Variations : Exploring Useful Information in Feature Maps by Its Variations”. In : (2021).
- [11] He Zhang Ren SUN. “Deep Residual Learning for Image Recognition”. In : (2015).
- [12] Minz WON, Keunwoo CHOI et Xavier SERRA. “Semi-Supervised Music Tagging Transformer”. In : (2021). arXiv : 2111.13457 [cs.SD].

Liste des tableaux

Table des figures

1	Répartition des classes dans le DataSet	4
2	Décompte des morceaux associés à un sous-genre unique <i>vs</i> plusieurs sous-genres	4
3	Nombre de combinaisons différentes de sous-genres par grande catégorie musicale	5
4	Correspondance entre échelles Mel et Hz	6
5	Spectrogramme Mel pour la catégorie 'pop'	7
6	Spectrogramme Mel pour la catégorie 'experimental'	7
7	Calcul des résidus profonds	8
8	Architecture du réseau RestNet50	8
9	Entraînement de la structure ResNet	9
10	Méthodologie de calcul des MFCC	10
11	Hélice de Shepard pour la perception de la hauteur	11
12	Chromagramme d'un morceau des Beatles	11
13	Point de Roll-Off à 95% sur le spectre d'amplitude	13
14	Diagrammes en violon réalisés sur le premier chroma et la moyenne spectrale du roll-off	14
15	Description graphique de l'algorithme KNN	15
16	Description graphique du SVM	15
17	Description graphique du Random Forest	16
18	Structure usuelle d'un perceptron multicouches	17
19	Méthode de Validation Croisée utilisée dans un GridSearch	18
20	Random Forest Feature Importance	18
21	Graphique de performances du réseau de neurones	19
22	Performances de Random Forest sur les différents sous-genres	19
23	Performances perceptron multicouche sur les différents sous-genres	20