**Rapport de projet**

Programme utilisant une intelligence artificielle pour transformer une musique en une partition pour guitare ou plus communément appeler tablature



Figure 1 : Logo application

*Projet réalisé par*

Yoann PRUDHOMME

*Projet encadrée par*

Mr Tomczak

**Résumé**

Ce projet, réalisé dans le cadre de mon projet de fin d’année de licence, consiste en une application capable de transformer un fichier audio en tablature de guitare. L’objectif est d’offrir aux musiciens un outil pratique pour transcrire automatiquement une mélodie sans avoir à le faire manuellement. L’application proposera une interface intuitive où l’utilisateur pourra importer un fichier audio au format MP3 ou WAV. Une fois le fichier sélectionné, le programme analysera le signal sonore en utilisant des techniques avancées de traitement du signal et d’intelligence artificielle. L’algorithme détectera les notes jouées, identifiera les différentes fréquences et les convertira en une tablature lisible pour guitare. L’application prendra en compte le style fingerstyle et cherchera à proposer une transcription la plus fidèle possible à l’original. Une fois l’analyse terminée, l’utilisateur pourra visualiser la tablature générée et l’exporter au format PDF pour une utilisation ultérieure. Ce projet vise à simplifier le travail des guitaristes en leur offrant un moyen rapide et efficace d’obtenir des partitions adaptées à leur instrument.

**Table des matières**

[I. INTRODUCTION 5](#_Toc191126808)

[1. Présentation du contexte 5](#_Toc191126809)

[2. Objectif du projet 5](#_Toc191126810)

[3. Défis rencontrés 6](#_Toc191126811)

[4. Résultats obtenus 6](#_Toc191126812)

[II. CAHIER DES CHARGES 7](#_Toc191126813)

[1. Problématique et besoins 7](#_Toc191126814)

[1) Besoins fonctionnels : 7](#_Toc191126815)

[2) Besoins non fonctionnels : 7](#_Toc191126816)

[2. Contraintes 7](#_Toc191126817)

[1) Techniques : 7](#_Toc191126818)

[2) Temporelles : 8](#_Toc191126819)

[3) Matérielles : 8](#_Toc191126820)

[3. Outils et technologies envisagés 8](#_Toc191126821)

[1) Bibliothèques et frameworks : 8](#_Toc191126822)

[2) Langages de programmation : 9](#_Toc191126823)

[3) Environnements de développement : 9](#_Toc191126824)

[4) Formats de sortie : 9](#_Toc191126825)

[III. CONCEPTION 9](#_Toc191126826)

[IV. IMPLEMENTATION 10](#_Toc191126827)

[1. Compréhension des modèles d’IA et premiers essais 10](#_Toc191126828)

[2. Recherche de datasets et première approche 10](#_Toc191126829)

[3. Difficultés avec le dataset et solutions 10](#_Toc191126830)

[4. Fonctionnement de l’IA 11](#_Toc191126831)

[1) Création du dataset 11](#_Toc191126832)

[a) Définition des notes et de leurs fréquences 11](#_Toc191126833)

[b) Prise en compte des variations naturelles 11](#_Toc191126834)

[c) Encodage des notes pour l’IA 11](#_Toc191126835)

[d) Normalisation des fréquences 12](#_Toc191126836)

[e) Séparation des données en entraînement et test 12](#_Toc191126837)

[2) Création du réseau de neurone pour utiliser se dataset 13](#_Toc191126838)

[a) Structure du réseau de neurones 13](#_Toc191126839)

[b) Compilation du modèle 13](#_Toc191126840)

[c) Mécanisme d'arrêt précoce (EarlyStopping) 14](#_Toc191126841)

[3) Entraînement du Modèle 14](#_Toc191126842)

[a) Processus d'entraînement 14](#_Toc191126843)

[b) Suivi des performances 14](#_Toc191126844)

[4) Prédiction d'une Note à partir d'une Fréquence 15](#_Toc191126845)

[a) Fonction de prédiction 16](#_Toc191126846)

[b) Étapes de la prédiction 16](#_Toc191126847)

[5. Détection des Fréquences dans un Fichier Audio 16](#_Toc191126848)

[1) Amélioration de l’Analyse grâce à la Transformée de Fourier 17](#_Toc191126849)

[2) Détection des Notes et Gestion des Intervalles 18](#_Toc191126850)

[3) Conversion en Tablature 18](#_Toc191126851)

[4) Enregistrement et Sauvegarde du Modèle 19](#_Toc191126852)

[V. TEST 20](#_Toc191126853)

[VI. CONCLUSION 20](#_Toc191126854)

[1. Démarche suivie 21](#_Toc191126855)

[2. Bilan du projet 21](#_Toc191126856)

[3. Perspectives d’évolution 22](#_Toc191126857)

[VII. ANNEXE 23](#_Toc191126858)

[VIII. WEBOGRAPHIE 23](#_Toc191126859)

# INTRODUCTION

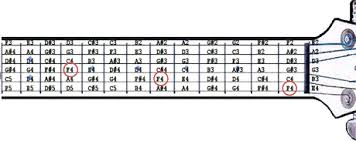
## Présentation du contexte

Étant en troisième année de licence informatique, je devais réaliser un projet de fin d’année. C’est justement à cette période que j’ai commencé à m’intéresser à l’intelligence artificielle. Passionné de guitare, j’ai constaté qu’il était souvent difficile d’obtenir des tablatures précises pour certaines musiques. C’est ainsi que m’est venue l’idée de concevoir un outil capable de générer automatiquement des tablatures à partir d’un enregistrement audio, en utilisant l’IA.

## Objectif du projet

L’objectif de ce projet est de développer une application permettant aux utilisateurs d’importer un fichier audio au format **.mp3** ou **.wav**, puis d’analyser son contenu pour en extraire les fréquences utiles. Ces fréquences sont ensuite transmises à une intelligence artificielle que j’ai entraînée afin de les convertir en notes de musique. Une fois les notes identifiées, une tablature est générée grâce à un tableau qui associe chaque note à une position sur le manche de la guitare.

Pour créer ce tableau, j’ai utilisé cette image :



## Défis rencontrés

Mon ambition initiale était de concevoir un programme capable d’analyser une musique complète, du début à la fin. Cependant, je me suis rapidement rendu compte que détecter précisément quelles notes étaient jouées et à quel moment représentait un défi bien plus complexe que prévu.

Ce projet est né d’un double intérêt : mon attrait pour l’IA et la guitare, ainsi que mon besoin personnel – partagé par d’autres musiciens – de trouver des tablatures précises sans devoir les transcrire manuellement. La création d’une tablature à l’oreille est une tâche fastidieuse pour ceux qui ne possèdent pas l’oreille absolue, une aptitude rare permettant d’identifier immédiatement une note sans référence. Si l’oreille relative peut être développée avec le temps, l’oreille absolue est souvent innée ou acquise très tôt.

## Résultats obtenus

À l’origine, je souhaitais que mon programme puisse analyser une musique complète et en extraire l’ensemble des notes. Toutefois, avec les outils et les méthodes que j’ai mis en place, j’ai constaté des limitations. Actuellement, mon programme parvient à détecter des notes jouées individuellement, mais il rencontre des difficultés lorsque les notes s’enchaînent trop rapidement. Il fonctionne donc de manière fiable si on lui fournit un enregistrement où les notes sont jouées une par une et lentement, comme lorsqu’on joue les six cordes d’une guitare séparément.

Ce projet représente une première étape vers une solution plus avancée, et il ouvre la voie à des améliorations futures pour mieux gérer les enchaînements de notes et la complexité des morceaux de musique.

# CAHIER DES CHARGES

## ****Problématique et besoins****

L'objectif est de développer une intelligence artificielle capable d'écouter un fichier audio (MP3 ou WAV) et de le convertir en tablature de guitare.  
**Problématique principale** : Comment extraire les notes et accords joués à partir d'un enregistrement audio, tout en respectant l'expressivité du jeu à la guitare (fingerstyle) ?

### ****Besoins fonctionnels :****

* Analyser un fichier audio pour détecter les notes jouées.
* Associer ces notes à des positions sur le manche de la guitare (tablature).
* Générer une tablature lisible et exportable (ex. PDF, format texte).

### ****Besoins non fonctionnels :****

* Une reconnaissance précise des notes, même en présence d’accords complexes.
* Un temps de traitement raisonnable pour l’utilisateur.
* Une interface simple d'utilisation.

## ****C****ontraintes

### ****Techniques :****

* Doit supporter les fichiers audio en **MP3 et WAV**.
* Précision de détection des notes et accords.
* Capacité à différencier plusieurs notes jouées simultanément (polyphonie).
* Gestion des bruits et variations (vibrato, bends).

### ****Temporelles :****

|  |  |  |
| --- | --- | --- |
| **Étape** | **Durée estimée** | **Délai** |
| Recherche bibliographique et documentation | 2 semaines | 29/10/2024 |
| Conception et entraînement d’un modèle CNN | 3 semaines | 19/11/2024 |
| Conception et entraînement du vrai modèle CNN | 3 semaines | 10/12/2024 |
| PAUSE EXAMEN | 1 semaine |  |
| Développement du traitement audio | 3 semaines | 25/01/2025 |
| Génération du fichier PDF | 3 jours | 28/01/2025 |
| Tests et validations | 1 semaine | 04/02/2025 |
| Rédaction du rapport final | 2 semaines | 18/02/2025 |

### ****Matérielles :****

* Besoin d’un ordinateur performant avec **GPU** si entraînement d’un modèle neuronal.

## ****Outils et technologies envisagés****

### ****Bibliothèques et frameworks :****

* **Librosa** 0.10.2.post1 (analyse audio et extraction de caractéristiques)
* **TensorFlow** 2.14.0 (modèle de machine learning)
* **Keras** 2.14.0 (API pour TensorFlow)
* **Scikit-Learn** 1.6.1 (prétraitement des données, modèles de base)
* **NumPy** 1.26.4 (manipulation des données audio)
* **SciPy** 1.13.1 (traitement du signal)
* **Matplotlib** 3.9.4 (visualisation des signaux et spectrogrammes)
* **FPDF** 1.7.2 (génération de la tablature en PDF)
* **Joblib** 1.4.2 (sauvegarde et chargement des modèles)
* **os / math** (bibliothèques standard de Python)
* Tous les **addons installés automatiquement** avec ces bibliothèques.

### ****Langages de programmation :****

* **Python** (langage principal)

### ****Environnements de développement :****

* **PyCharm** (Environnement python de jetbrain toolbox)

### ****Formats de sortie :****

* PDF

# CONCEPTION

L'organisation du projet repose sur une structure modulaire permettant de séparer clairement les différentes étapes du traitement, de l’analyse du son jusqu’à la génération de tablatures de guitare.

Tout d’abord, les fichiers sont organisés en plusieurs dossiers afin de bien différencier les données, le code et les résultats. Le dossier **"Son"** contient les fichiers audio qui seront analysés. Le dossier **"Modele IA"** stocke les modèles d’intelligence artificielle utilisés pour la reconnaissance des notes. Le dossier **"PositionNote"** contient un fichier excel ainsi qu’une image qui permette de mettre en correspondance une note avec une position sur le manche manuellement.

Ensuite, le dossier **"Tablature" contient la tablature créer du son en paramètre**. Le script **"DetecterLaFrequenceJouer.py"** analyse les fichiers audio et extrait les notes jouées, **"CreationDuTableau.py"** organise ces notes sous une forme exploitable pour une tablature. Le script **"GenerationDeLIA.py"** permet de crée le modèle d’IA, tandis que **"main.py"** centralise l’exécution de l’ensemble du processus.

Cette structure modulaire permet une meilleure organisation du projet, facilitant la maintenance et l’évolution du code. Elle sépare clairement les différentes étapes du traitement et assure une plus grande lisibilité du projet, en rendant chaque module indépendant et réutilisable.

# IMPLEMENTATION

## ****Compréhension des modèles d’IA et premiers essais****

Avant de me lancer dans ce projet, j’ai d’abord cherché à comprendre le fonctionnement des réseaux de neurones convolutionnels en suivant des vidéos YouTube détaillant chaque paramètre modifiable. Pour expérimenter, j’ai créé mon premier modèle d’IA capable de différencier une image de chien d’une image de chat.

Une fois cette première expérience réussie, j’ai voulu adapter cette approche aux sons. Mon objectif était de développer une intelligence artificielle capable d’analyser un fichier audio (MP3 ou WAV) et d’en extraire les notes de guitare sous forme de tablature.

## ****Recherche de datasets et première approche****

J’ai commencé par chercher un dataset contenant des notes et des accords de guitare. Le meilleur dataset que j’ai trouvé contenait uniquement des notes individuelles, et non des accords complexes. J’ai décidé que si mon projet fonctionnait bien avec ce dataset, je chercherais soit un autre dataset plus adapté, soit je créerais mon propre jeu de données.

J’ai ensuite développé un programme capable de prendre un son en entrée (WAV ou MP3 via la bibliothèque **Librosa**) et de ressortir une note exploitable par l’IA. L’objectif était que l’IA, entraînée sur un ensemble de notes, puisse reconnaître la note jouée en sortie.

## ****Difficultés avec le dataset et solutions****

Après plusieurs essais avec le dataset trouvé en ligne, j’ai rencontré des difficultés et j’ai finalement décidé d’adopter une autre approche :

* J’ai généré moi-même des données en utilisant des fréquences précises pour chaque note, en les modifiant légèrement (quelques Hertz) pour généraliser l’apprentissage du modèle.

Malgré cela, mon IA ne fonctionnait toujours pas correctement. Après plusieurs tests et réflexions, j’ai finalement demandé à ChatGPT de m’aider à identifier l’erreur. Il s’est avéré que je n’avais **pas normalisé mes données**.

* Normaliser les données signifie transformer les valeurs pour qu’elles se situent dans une plage entre **0 et 1 exclu**, ce qui est essentiel pour assurer un bon apprentissage du modèle.

Après avoir corrigé cela, mon IA a enfin commencé à fonctionner comme prévu : elle prenait une fréquence en entrée et ressortait la note correspondante avec succès.

## Fonctionnement de l’IA

### Création du dataset

L'intelligence artificielle développée repose sur l'analyse des fréquences sonores pour reconnaître les notes de musique. Voici comment elle fonctionne :

#### **Définition des notes et de leurs fréquences**

Chaque note de musique correspond à une fréquence spécifique en Hertz (Hz). Par exemple, la note A2 a une fréquence d’environ 110 Hz, la note B3 est autour de 247 Hz, etc. Un dictionnaire a été créé pour associer chaque note à sa fréquence.

#### Prise en compte des variations naturelles

Dans un enregistrement réel, les fréquences ne sont jamais parfaitement précises en raison des variations liées à l’instrument ou aux conditions d’enregistrement. Pour tenir compte de ces écarts, j’ai ajouté une marge de ±5 Hz aux fréquences des notes, ce qui aide l’IA à mieux reconnaître les notes, même si elles ne sont pas parfaitement accordées.

#### Encodage des notes pour l’IA

L’IA ne comprend pas les noms des notes (A2, B3, C4…). Elle fonctionne uniquement avec des nombres. Pour cela, j’utilise un procédé appelé **Label Encoding**, qui transforme chaque note en un nombre unique :

* + A2 → 0
  + B3 → 1
  + C4 → 2
  + D5 → 3

Ensuite, pour que l’IA puisse facilement différencier les notes, j’applique une méthode appelée **One-Hot Encoding**. Cette méthode consiste à créer un tableau où chaque note est représentée par une série de 0, sauf pour la position correspondant à la note actuelle, qui est mise à 1 :

| **Note** | **Encodage** | **One-Hot Encoding** |  |
| --- | --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| Note | Encodage | One-hot encoding |
| A2 | 0 | [1,0,0,0] |
| B3 | 1 | [0,1,0,0] |
| C4 | 2 | [0,0,1,0] |
| D5 | 3 | [0,0,0,1] |

Grâce à cet encodage, l’IA peut facilement identifier chaque note et lui attribuer une probabilité en fonction de la fréquence détectée.

#### Normalisation des fréquences

Les fréquences des notes sont ensuite **normalisées** pour faciliter l’apprentissage du modèle. La normalisation consiste à ramener toutes les valeurs dans un intervalle entre 0 et 1, ce qui permet à l’IA de mieux interpréter les données.

Pour cela, j’utilise la fonction **MinMaxScaler** de la bibliothèque **scikit-learn**. Elle fonctionne en trouvant la fréquence la plus élevée et la plus basse, puis en recalculant toutes les autres valeurs sous forme de pourcentage entre ces deux extrêmes. Cela évite que certaines notes aient plus d’influence que d’autres simplement parce qu’elles ont une fréquence plus élevée.

#### Séparation des données en entraînement et test

Pour que l’IA apprenne à reconnaître les notes, je dois lui fournir des exemples. Je divise donc les données en deux groupes :

* + **80 % des données** servent à entraîner le modèle (l’IA apprend à reconnaître les notes).
  + **20 % des données** servent à tester le modèle après l’entraînement, pour vérifier s’il est capable d’identifier correctement les notes qu’il n’a jamais vues auparavant.

En appliquant ces étapes, l’IA est capable de comprendre les fréquences sonores et de reconnaître les notes de musique avec précision.

### ****Création du réseau de neurone pour utiliser se dataset****

Pour reconnaître les notes de musique à partir des fréquences, j’ai construit un **réseau de neurones** en utilisant la bibliothèque **Keras**. Ce réseau est un modèle d’intelligence artificielle qui apprend à associer une fréquence à la bonne note grâce à plusieurs couches de neurones.

#### Structure du réseau de neurones

Le modèle est composé de plusieurs couches :

* **Couche d’entrée** : Cette couche contient **1 seul neurone**, qui reçoit en entrée une fréquence normalisée (une valeur entre 0 et 1).
* **Couches cachées** :
  + Trois couches intermédiaires sont utilisées avec **128, 256 et 128 neurones** chacune.
  + Elles utilisent une fonction d’**activation ReLU** (Rectified Linear Unit), qui permet au modèle d’apprendre des relations complexes entre les fréquences et les notes.
  + Un **Dropout de 0.2** est appliqué après chaque couche cachée. Cette technique désactive **aléatoirement 20 % des neurones** à chaque étape d’apprentissage, ce qui évite que le modèle ne mémorise trop les données et améliore sa capacité à généraliser sur de nouvelles notes.
* **Couche de sortie** :
  + Cette couche contient **un nombre de neurones égal au nombre total de notes à reconnaître**.
  + Chaque neurone représente une note et utilise une **activation Softmax**, qui transforme les scores calculés par le réseau en **probabilités**. Cela permet au modèle de donner une estimation de la note la plus probable.

#### Compilation du modèle

Avant de lancer l’apprentissage, le modèle est **compilé**, c'est-à-dire préparé avec des paramètres spécifiques :

* **Optimiseur Adam** : Il s'agit d'un algorithme qui ajuste les poids du réseau pour améliorer sa précision à chaque itération. Adam est un optimiseur efficace et largement utilisé en apprentissage profond.
* **Fonction de perte categorical\_crossentropy** : Cette fonction mesure l’écart entre les prédictions du modèle et les notes réelles, en attribuant un score de qualité à chaque estimation. L’objectif du modèle est de **minimiser cette perte** pour améliorer ses performances.

#### Mécanisme d'arrêt précoce (EarlyStopping)

Pour éviter que l’IA **ne continue à s’entraîner inutilement** lorsque ses performances n’évoluent plus, j’utilise une technique appelée **EarlyStopping**.

* Ce mécanisme surveille la **précision de l’IA sur les notes test** à chaque époque (un cycle d’apprentissage).
* Si, après **20 époques**, la précision ne s’améliore plus, l’entraînement est automatiquement **arrêté**.
* Cela empêche le modèle de **surcharger son apprentissage** et lui permet d’être plus efficace sur de nouvelles données.

Grâce à cette architecture, le réseau de neurones peut analyser les fréquences et reconnaître les notes de manière précise tout en évitant les erreurs dues au surapprentissage.

### ****Entraînement du Modèle****

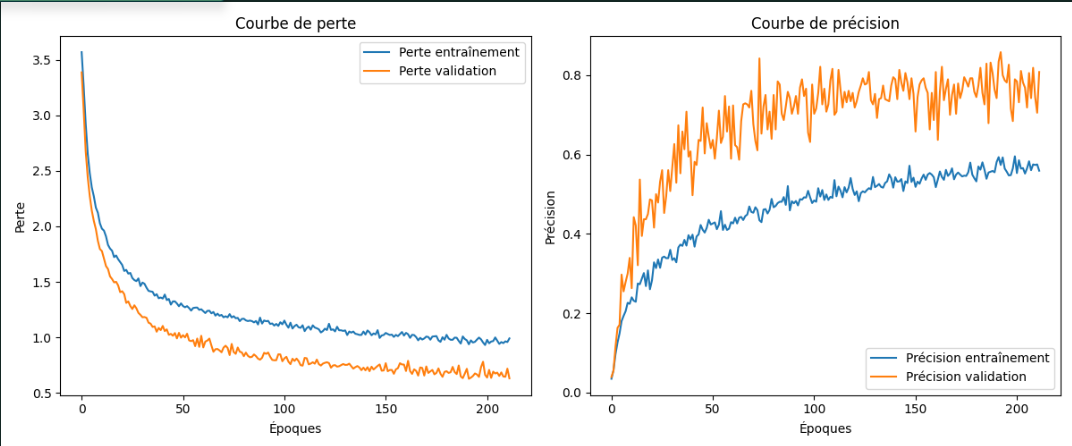
L'entraînement du modèle est une étape essentielle pour permettre à l'intelligence artificielle de reconnaître les notes de musique avec précision.

#### Processus d'entraînement

* Le modèle est entraîné sur **500 époques**.
* À chaque époque, l'IA ajuste ses paramètres pour améliorer la reconnaissance des notes.
* Un **ensemble de test** est utilisé pour valider l’apprentissage et vérifier si le modèle généralise bien sur de nouvelles données.
* Ce modèle n'est **pas très complexe** (il n’a qu’une seule entrée et une seule sortie), donc 500 époques suffisent.

#### Suivi des performances

Pour visualiser l’évolution de l’entraînement, une fonction appelée **tracer\_historique()** permet d'afficher **deux graphiques** :



* **Courbe de perte** (graphique de gauche) :
  + Elle montre les erreurs du modèle au fil des époques.
  + Deux courbes apparaissent :
    - La **courbe bleue** correspond aux erreurs faites sur les données d’apprentissage.
    - La **courbe orange** correspond aux erreurs sur les données de test.
  + L’objectif est d’avoir une perte qui diminue progressivement, signe que l’IA apprend bien.
* **Courbe de précision** (graphique de droite) :
  + Elle montre à quel point le modèle reconnaît correctement les notes.
  + Deux courbes sont présentes :
    - **La courbe bleue représente l’apprentissage** (précision sur les données d'entraînement).
    - **La courbe orange représente les tests** (précision sur les nouvelles données).
  + On peut voir qu'après un certain nombre d’époques, la précision **se stabilise autour de 0.8 (80%)**, ce qui indique que le modèle reconnaît correctement la majorité des notes.

Grâce à ces graphiques, il est possible de suivre l’évolution du modèle et de s'assurer qu’il apprend efficacement sans trop d’erreurs.

### Prédiction d'une Note à partir d'une Fréquence

L'objectif de cette étape est de **convertir une fréquence sonore en une note de musique** en utilisant le modèle de réseau de neurones. Grâce à l'apprentissage effectué précédemment, l'IA est capable d'identifier la note la plus probable correspondant à une fréquence donnée.

#### Fonction de prédiction

L'IA utilise une fonction spécifique qui prend une fréquence en entrée et renvoie la note de guitare correspondante. Cette fonction repose sur le **modèle entraîné** et sur sa **couche de sortie softmax**, qui permet d’attribuer une probabilité à chaque note.

#### Étapes de la prédiction

1. **Analyse de la fréquence d’entrée**
   * La fréquence est d'abord **normalisée**, c'est-à-dire convertie en une valeur entre 0 et 1 pour être compatible avec le modèle.
   * Elle est ensuite transformée sous une forme exploitable par le réseau de neurones et transmise au modèle, qui va analyser son contenu.
   * Le modèle génère en sortie un **ensemble de probabilités**, où chaque probabilité est associée à une note.
2. **Interprétation des résultats**
   * La **couche de sortie softmax** ajuste ces probabilités de manière à ce que leur somme soit égale à 1.
   * Plus une probabilité est élevée, plus la fréquence analysée a de chances de correspondre à cette note précise.
   * L'IA repère donc **la note qui possède la probabilité la plus forte**.
3. **Conversion en note musicale**
   * L’indice correspondant à la plus haute probabilité est **converti en une note**.
   * Pour cela, l’IA utilise une table de correspondance établie lors de l'entraînement, qui permet d’associer chaque valeur numérique à une note spécifique comme A2, B3, C4, etc.
4. **Affichage du résultat**
   * L’IA renvoie alors **le nom de la note détectée**, qui est celle jugée la plus probable en fonction des données analysées.

Grâce à ce processus, l’IA est capable d’identifier et d’afficher la note correspondant à une fréquence donnée.

## Détection des Fréquences dans un Fichier Audio

L'objectif de cette étape est d'extraire les fréquences principales d'un fichier audio en analysant son spectre fréquentiel. Pour cela, la bibliothèque Librosa est utilisée afin de réaliser une analyse approfondie du signal sonore et d’identifier les fréquences correspondant aux notes jouées.

L’analyse commence par le chargement du fichier audio avec une fréquence d'échantillonnage de 11025 Hz, un choix qui permet d’optimiser la détection des fréquences tout en réduisant la complexité du traitement. Une fois le signal chargé, il est converti en un spectrogramme à l'aide de la Transformation de Fourier à Court Terme (STFT). Cette transformation permet d’obtenir une représentation du signal dans le domaine fréquentiel, grâce à une fenêtre de 2048 points et un pas de 20 millisecondes, offrant ainsi une analyse détaillée des variations de fréquence dans le temps.

Pour extraire les fréquences dominantes, le signal est segmenté en intervalles de 5 millisecondes. À chaque intervalle, les amplitudes moyennes sont calculées afin de détecter les pics d’amplitude, qui correspondent aux fréquences les plus marquées du signal. Cependant, toutes ces fréquences ne sont pas forcément pertinentes pour l’analyse musicale, car le son d’une guitare produit non seulement une fréquence fondamentale mais aussi plusieurs harmoniques et bruits parasites.

Afin de ne conserver que les informations utiles, un filtrage est appliqué. Les fréquences détectées sont analysées et comparées pour identifier celles qui correspondent réellement à des notes de guitare. Seules celles comprises entre 70 Hz et 710 Hz sont retenues, car elles correspondent à la plage des notes jouables sur un manche de guitare standard. Cela permet d’éliminer les bruits parasites et d’affiner la précision de la détection.

À l’issue de cette analyse, une liste des fréquences principales est obtenue. Ces données serviront ensuite de base pour la conversion en tablature, permettant ainsi de retranscrire fidèlement la mélodie capturée dans l’enregistrement audio.

### Amélioration de l’Analyse grâce à la Transformée de Fourier

En explorant différentes approches, j’ai constaté que l’analyse fréquentielle pouvait être optimisée grâce à une segmentation plus fine du signal. Une note de guitare ne produit pas seulement une fréquence fondamentale, mais également plusieurs harmoniques, ce qui complique la détection des notes réelles. De plus, certains bruits parasites, situés en dehors de la plage audible de la guitare, doivent être filtrés pour améliorer la précision de l’analyse.

En discutant avec des amis, une approche plus précise m’a été suggérée. Plutôt que d’afficher directement l’évolution des fréquences sur toute la durée du signal, ce qui manque souvent de précision, il est plus efficace de diviser le signal en petites fenêtres temporelles, d’environ 25 millisecondes. Cette segmentation permet d’extraire les fréquences dominantes dans chaque intervalle et d’améliorer ainsi la détection des notes.

Grâce à cette méthode, il devient possible de générer un spectrogramme détaillé qui met en évidence les fréquences importantes. Cela facilite non seulement l’identification des notes jouées, mais aussi la distinction entre les harmoniques et les sons parasites, rendant ainsi l’analyse plus fiable et plus précise.

### Détection des Notes et Gestion des Intervalles

Une fois la détection des fréquences mise en place, un nouveau défi est apparu. Une musique contient un très grand nombre de fenêtres d’analyse, ce qui complique la détermination du moment précis où une note est réellement jouée. De plus, la présence de bruits parasites, d’harmoniques et des variations naturelles du son rend cette tâche encore plus complexe.

Mon programme parvient à détecter une suite de notes, mais il présente encore certaines limites. Il fonctionne bien lorsque les notes ne sont pas jouées trop rapidement et lorsque leurs fréquences sont suffisamment distinctes. En revanche, il a plus de difficulté à distinguer des notes très rapprochées ou jouées à une vitesse élevée.

L’intelligence artificielle intervient ensuite pour associer chaque fréquence détectée à la note la plus probable parmi celles sur lesquelles elle a été entraînée. Cette étape permet d’affiner les résultats et d’améliorer la précision de la transcription en tablature.

### ****Conversion** **en Tablature****

La transformation des notes détectées en tablature suit plusieurs étapes :

1. **Détection des notes** :  
   Après l'analyse du fichier audio, les notes jouées sont identifiées grâce à un modèle d'intelligence artificielle. Chaque note détectée est associée à une fréquence spécifique, prédite par le modèle.
2. **Association aux positions sur le manche** :  
   Pour chaque note reconnue, un dictionnaire associant les notes aux positions sur le manche de la guitare est utilisé. Ce dictionnaire indique la corde et la case à jouer pour chaque note.
3. **Génération de la tablature** :  
   Les positions (corde et case) correspondant aux notes détectées sont ajoutées à une structure de données représentant la tablature.
4. **Mise en forme de la tablature** :  
   Une fois toutes les notes transcrites, la tablature est organisée pour être lisible, en disposant les notes sur les lignes représentant les cordes de la guitare.
5. **Affichage et sauvegarde** :  
   La tablature finale peut être affichée à l'écran ou exportée sous forme de fichier PDF. Cette sauvegarde permet aux guitaristes de rejouer la mélodie transcrite.

Grâce à ce processus, la conversion de l'audio en tablature facilite l’interprétation des morceaux enregistrés et leur restitution sur un instrument.

### Enregistrement et Sauvegarde du Modèle

Lorsqu'un modèle est entraîné et validé, plusieurs éléments sont sauvegardés pour garantir sa réutilisation :

1. **Modèle entraîné** :  
   Le réseau de neurones est sauvegardé sous forme d’un fichier .h5, conservant à la fois son architecture et ses poids.
2. **Scaler des fréquences** :  
   Pour assurer une cohérence dans le traitement des données, le scaler utilisé pour normaliser les fréquences d’entrée est enregistré avec joblib.
3. **Encodeur des étiquettes** :  
   L’encodeur, qui transforme les noms des notes en valeurs numériques et inversement, est sauvegardé pour garantir la bonne reconversion des prédictions en notes lisibles.

Ces fichiers assurent la pérennité du modèle et permettent son utilisation sur de nouvelles données sans nécessiter un nouvel entraînement.

# **TEST**

Pour réaliser mes tests, j’utilise ma propre guitare ainsi qu’un micro branché directement afin de limiter les bruits parasites et obtenir un son plus propre. Cela me permet d’effectuer des tests variés, en jouant des notes isolées, des suites mélodiques et en variant l’intensité du jeu.

J’augmente progressivement la difficulté en testant des tempos plus rapides ou plus lent. De cette manière, je peux évaluer la précision de la détection et identifier les situations où le programme rencontre des difficultés.

Ces tests me permettent également d’ajuster les paramètres du modèle et d’améliorer la qualité des prédictions avant d’envisager une généralisation à des morceaux plus complexes.

# **CONCLUSION**

Ce projet m’a permis d’explorer les défis du traitement du son et de l’apprentissage automatique appliqué à la musique. J’ai suivi une démarche progressive pour construire une

intelligence artificielle capable de transformer un enregistrement audio en une tablature lisible.

## Démarche suivie

1. **Étudier les modèles d’IA** : J’ai d’abord travaillé sur la classification d’images pour me familiariser avec les concepts fondamentaux des réseaux de neurones et des modèles d’apprentissage supervisé. Cette première approche m’a permis de comprendre comment structurer un modèle d’analyse et d’extraction de caractéristiques.
2. **Explorer les datasets et construire un modèle pour reconnaître les notes** : J’ai constitué un jeu de données adapté en organisant des échantillons sonores correspondant à différentes notes de guitare. J’ai ensuite entraîné un modèle pour reconnaître ces notes en fonction de leurs caractéristiques fréquentielles.
3. **Améliorer l’approche avec Librosa** : Librosa m’a permis d’extraire plus précisément les fréquences dominantes des sons enregistrés, en affinant la détection grâce à des techniques avancées de traitement du signal. Cela a été une étape clé pour fiabiliser les résultats et mieux interpréter les sons de la guitare.
4. **Développer un programme de détection des notes à partir des fréquences détectées**: J’ai conçu un programme permettant d’identifier les fréquences extraites et de les associer aux notes les plus probables. Cette étape a nécessité des ajustements pour tenir compte des écarts de fréquence pouvant apparaître en fonction de l’instrument et du jeu du guitariste.
5. **Faire en sorte que l’IA associe ces fréquences à des notes spécifiques** : Enfin, j’ai mis en place un système de classification qui traduit les fréquences en notes identifiables, avec une approche permettant d’adapter le modèle aux variations naturelles du jeu de guitare.

## Bilan du projet

Ce projet m’a apporté une compréhension approfondie des défis liés à la détection des notes à partir d’un enregistrement audio. À ce stade, mon programme est capable de détecter une suite de notes distinctes, tant qu’elles ne sont pas jouées trop rapidement et que leurs fréquences ne se chevauchent pas excessivement. L’IA parvient à identifier et à associer ces notes à la bonne catégorie avec une précision satisfaisante dans des conditions contrôlées.

En revanche, certaines limites restent à surmonter. Par exemple, la détection devient plus difficile lorsque les notes sont enchaînées rapidement ou lorsque plusieurs fréquences coexistent, ce qui est fréquent dans des morceaux plus complexes. L’un des défis majeurs est donc d’améliorer la robustesse du modèle face à ces situations.

## Perspectives d’évolution

L’objectif à moyen terme est d’affiner le modèle pour améliorer sa précision et sa capacité à traiter des morceaux plus riches et expressifs. Plusieurs pistes d’amélioration peuvent être envisagées :

* **Optimisation du modèle d’apprentissage** : Entraîner le modèle sur un dataset plus varié pour mieux gérer les variations de jeu et les différences entre guitares.
* **Gestion des notes rapides et des transitions** : Intégrer des techniques plus avancées pour détecter les enchaînements rapides et affiner la segmentation des notes dans un enregistrement.
* **Prise en charge de la polyphonie** : Explorer des solutions pour identifier plusieurs notes jouées simultanément, ce qui permettrait à l’IA d’analyser des morceaux plus complexes incluant des accords.
* **Affichage et interaction** : Développer une interface utilisateur qui permettrait de visualiser la tablature en temps réel et d’interagir avec les résultats pour les modifier ou les ajuster si nécessaire.
* **Génération automatique de tablatures adaptées** : Aller au-delà de la simple reconnaissance de notes en proposant des doigtés optimisés pour la guitare, facilitant ainsi l’apprentissage et l’interprétation des morceaux.

En conclusion, ce projet représente une première étape encourageante vers la génération automatique de tablatures à partir de l’audio. Même si certains défis techniques restent à relever, les résultats obtenus sont prometteurs et ouvrent la voie à des améliorations qui pourraient rendre cet outil plus performant et accessible aux musiciens.

# ANNEXE

**Github** : https://github.com/Loobixx/Projet-AI/tree/master

# WEBOGRAPHIE

[**https://youtu.be/XUFLq6dKQok?si=muZN1128aenKZt1M**](https://youtu.be/XUFLq6dKQok?si=muZN1128aenKZt1M)

Voici une série de vidéo YouTube au départ que j’ai regardé pour apprendre comment fonctionnais une IA.

[**https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://moodle.luniversitenumerique.fr/mod/resource/view.php%3Fid%3D562&ved=2ahUKEwiiiMf-qNeLAxW4RKQEHZhmMAsQFnoECBoQAQ&usg=AOvVaw0i98B38-TP16Vy02MI5yRd**](https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://moodle.luniversitenumerique.fr/mod/resource/view.php%3Fid%3D562&ved=2ahUKEwiiiMf-qNeLAxW4RKQEHZhmMAsQFnoECBoQAQ&usg=AOvVaw0i98B38-TP16Vy02MI5yRd)

Ce document de l’INSA Toulouse m’a permis de découvrir la transformée de fourrier en détail et principalement le domaine fréquentiel que j’utilise dans mon programme