



école supérieure de  
génie informatique

---

## [RAPPORT PROJET ANNUEL]

---

[1.0]

[26/06/2023]

[ BRONGNIART Arthur | BOUDON Alexandre | YE Steven ]  
2022 – 2023

# Sommaire

<b>Introduction.....</b>	<b>3</b>
Contexte.....	3
Objectifs.....	4
<b>Description fonctionnelle.....</b>	<b>5</b>
Veille fonctionnelle sur l'existant.....	5
Aperçu des contenus.....	6
Usage.....	10
<b>Architecture technique.....</b>	<b>11</b>
Etat de l'art.....	11
Briques technologiques.....	14
Choix architecturaux.....	16
<b>Données mises en œuvre.....</b>	<b>17</b>
Récupération de la donnée.....	17
Traitement de la donnée.....	18
Evolution de la donnée.....	20
<b>Algorithmes mis en œuvre.....</b>	<b>21</b>
Algorithme de scrap et de traitement.....	21
Algorithme d'entraînement.....	22
Job kubernetes.....	27
<b>Gestion de projet.....</b>	<b>27</b>
Planning de réalisation.....	27
Répartition du travail.....	28
Problèmes rencontrés.....	28
<b>Résultats obtenus.....</b>	<b>31</b>
Bilan fonctionnel.....	31
Améliorations possibles.....	31

# Introduction

## Contexte

Avec l'évolution des plateformes vidéos, de nombreux artistes amateurs comme professionnels ont publié leurs réalisations musicales accessibles à tous. En parallèle de cette augmentation d'artistes, de nombreux musiciens amateurs ont voulu réinterpréter la mélodie présente dans ces musiques. Malheureusement, les partitions de musiques ne sont pas autant mis à disposition que les musiques de ces artistes.

Afin de combler en partie ce manque de partitions, plusieurs bénévoles mettent à disposition des partitions créées par leurs soins sur des sites communautaires. Très souvent transcrits par une personne ayant l'oreille absolue, celles-ci peuvent contenir des défauts et leurs nombres restent limitées.

Avec l'arrivée de nombreux logiciels axés sur la musique, un nouveau format de partition est apparu, les fichiers MIDI (format .mid). Ce fichier est une partition numérique pouvant être lue par un logiciel et pouvant inclure un ou plusieurs instruments.

Avec l'arrivée de cette technologie, de nombreux fichiers MIDI sont apparus sur le web disponibles pour tous. Cependant, au même titre que les partitions classiques, celles-ci peuvent contenir des erreurs et ne couvrent pas l'ensemble des musiques existantes.

Afin de répondre à la fois à la problématique du nombre réduit de fichiers MIDI et de la qualité de ces fichiers, un nouvel outil a été mis en place à disposition de n'importe quel utilisateur: OrchestrAI.

OrchestrAI est un site web permettant la transcription d'une musique youtube en fichier MIDI (une partition électronique). Cette génération est possible grâce à un modèle d'IA supervisé se basant sur plusieurs papiers scientifiques. Ce modèle est régulièrement mis à jour automatiquement avec toutes les musiques demandées par les utilisateurs.

Afin de réaliser ce projet de génération, nous avons formé un groupe de 3 étudiants où chacun a pu apporter des connaissances et des expériences dans le domaine de l'IA et de la data :

- Alexandre Boudon, dont la force repose sur la data, le scraping, l'utilisation et la mise en place de plateformes cloud et la création du front
- Arthur Brongniart, dont la qualité est d'être hybride, pouvant travailler à la fois sur l'infrastructure cloud ainsi que la conception et l'apprentissage de modèle
- Steven Ye, dont la spécialité est la data science, l'apprentissage de modèle et la résolution de problématiques liées à ceci

À travers ce projet, nous avons voulu mettre à profit les différentes compétences acquises durant ce Mastère en Intelligence Artificielle et Big Data et valider notre diplôme avec les spécialisations et expertises de chaque étudiant.

Nous avons choisi la génération de fichier MIDI par apprentissage supervisé en utilisant la plateforme cloud Azure, un site web front en php et une automatisation au lancement et à l'apprentissage grâce à Kubernetes, Docker et des scripts python.

## Objectifs

Afin de répondre à la problématique principale qui était de mettre à disposition une plateforme prenant en charge notre modèle, plusieurs objectifs ont été fixés afin de considérer celui-ci comme abouti:

- Pouvoir proposer un modèle généraliste pouvant prendre en entrée à la fois de la pop mais aussi du jazz ou du classique et arriver à pouvoir recréer la mélodie ou au mieux la musique de manière précise,
- Proposer une solution à faible coût et un entretien limité pour la personne en charge de la plateforme cloud,
- Proposer une plateforme simple et intuitive d'utilisation pour les différents utilisateurs, peu importe leur niveau dans le domaine musical.

# Description fonctionnelle

## Veille fonctionnelle sur l'existant

À ce jour, il existe plusieurs approches concernant la génération de musiques. Nous pouvons l'apercevoir à travers plusieurs plateformes comme Chordify, proposant un service de transcription de vidéos youtube en fichiers MIDI en ligne. Malgré ses avantages de disponibilité (web et application), l'utilisateur se retrouve rapidement restreint lorsqu'il souhaite utiliser ses propres fichiers ou liens musicaux.

The screenshot shows the Chordify website interface. At the top, there's a search bar, a file upload button, and navigation links for 'Découvrir', 'Créer un compte', 'Tarifs', 'Blog', 'Setlists', 'Aide', and more. A prominent 'Upgrade' button is visible. Below the header, a modal window encourages users to upload their own files for premium features. The main content area displays a song page for 'Paul McCartney & Wings - Mull Of Kintyre'. It includes a player toolbar with controls for play/pause, volume, tempo, capo, transposer, midi export, and printing. The song title and artist are shown above the player. Below the player is a guitar tablature with five frets and strings labeled 1, 2, 3, 4, 5. Fret 1 has 'X O' and 'O' markers. Fret 2 has 'XX O' and 'O' markers. Fret 3 has 'X O' and 'O' markers. Fret 4 has 'XX O' and 'O' markers. Fret 5 has 'X O' and 'O' markers. The notes are labeled 'La', 'Re', 'La', 'Re', 'La'. To the right of the tablature is a video thumbnail for the song. On the far right, there's a sidebar with a list of recommended songs from the 'Starter Songs' playlist, including R.E.M., Taylor Swift, Bob Marley, Paul McCartney & Wings, Bob Marley & The Wailers, and Elvis Presley, each with a small thumbnail and a link to the original source.

## Aperçu des contenus

Le site est composé de 4 interfaces dont 3 sont entièrement consacrées à l'utilisateur. Une partie de celles-ci nécessite une connexion de la part de l'utilisateur pour accéder aux services du site.

La première interface est la page d'accueil, l'utilisateur aura la possibilité de se connecter et de consulter les derniers fichiers MIDI générés par la communauté. Un mode sombre est disponible.

Accueil Conversion Support

Mode sombre

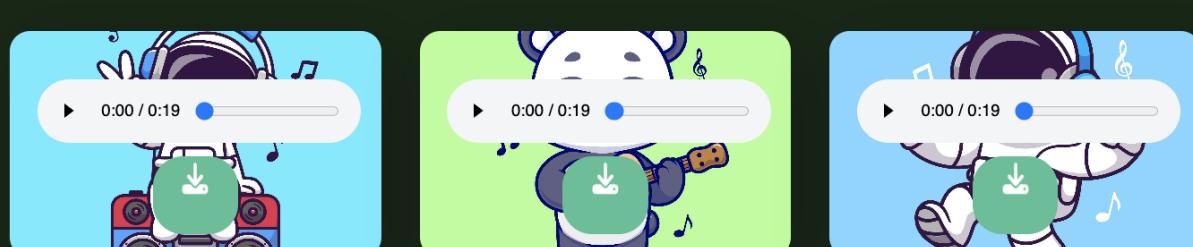


**Une conversion ? Besoin d'une partition ?**

Ce site a pour but de convertir des vidéos Youtube ou des fichiers MP3 en partition MIDI.

[CONNEXION](#) [INSCRIPTION](#)

**Une communauté active**



**NAVIGATION**

Conversion

Support

**A PROPOS**

OrchestrAI est un site de conversion d'un audio en fichier MIDI à partir d'une IA

OrchestrAI est un projet scolaire d'étudiants spécialisés en Intelligence Artificielle et Big DATA.

La seconde interface est la page de conversion : l'utilisateur est invité à entrer un lien youtube et à appuyer sur le bouton "Démarrer", après une courte attente, un fichier sera généré et pourra être visualisé et téléchargé.

The screenshot shows the 'Conversion' page. At the top, there is a navigation bar with 'Accueil', 'Conversion', and 'Support' buttons. To the right of the navigation bar are icons for brightness, a user profile, and a gear. Below the navigation bar, the main content area has a dark background with a decorative illustration of a character with blue hair and a golden staff on the right side. The central text reads 'Conversion de la musique'. Below it, there is a section titled 'Fournissez la musique' with a 'Lien youtube' input field containing a red YouTube icon and the text 'Lien youtube'. A green 'DÉMARRER' button is positioned below the input field. At the bottom right of the main area, there is a blue link labeled 'Voir vos conversion'.

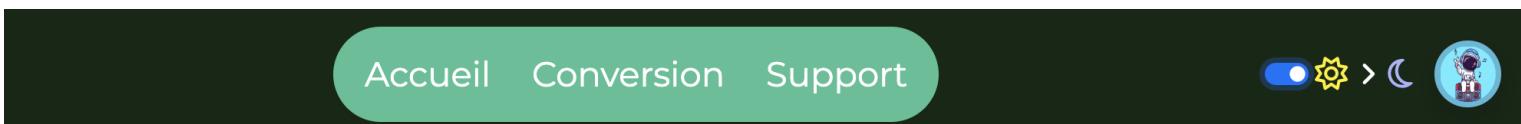
The screenshot shows the footer section. On the left, under 'NAVIGATION', there are links for 'Conversion' (with a circular arrow icon) and 'Support' (with a gear icon). On the right, under 'A PROPOS', there is information about the site: 'OrchestrAI est un site de conversion d'un audio en fichier MIDI à partir d'une IA' and 'OrchestrAI est un projet scolaire d'étudiants spécialisés en Intelligence Artificielle et Big DATA.'

L'autre partie de la seconde interface est la page des résultats de conversion. Après le temps de conversion, l'utilisateur aura accès à une nouvelle section où il pourra consulter le fichier généré de plusieurs manières : l'écouter, le visualiser ou le télécharger. L'utilisateur pourra juger de la qualité du fichier généré avant même de l'utiliser.

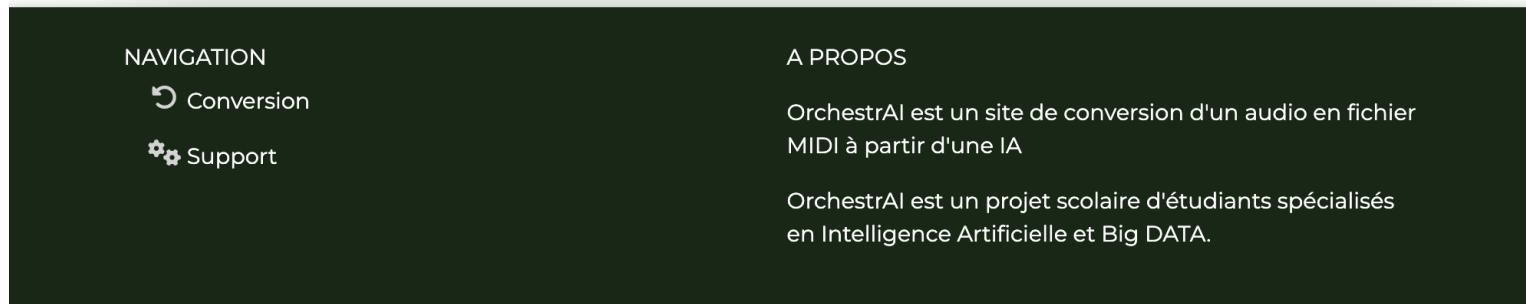
The screenshot shows a user interface for a file conversion tool. At the top, there is a navigation bar with three tabs: "Accueil" (Home), "Conversion" (Conversion), and "Support". To the right of the tabs are icons for brightness and a user profile. The main content area features a dark background with a stylized illustration of a character with blue hair and a golden horn, surrounded by swirling flames. To the right of the illustration, the text "Conversion réussie !!" (Conversion successful!!) is displayed above a waveform visualization. Below the waveform is a progress bar showing "0:00 / 0:00". At the bottom right, there is a button labeled "TÉLÉCHARGER" with a download icon.

La dernière interface utilisateur est la page d'historique des fichiers MIDI, l'utilisateur y trouvera l'intégralité de ses conversions triées par date de création. Il aura la possibilité d'écouter de nouveau le fichier MIDI avant de le télécharger ou de le supprimer.

L'interface de conversion et l'historique nécessite que l'utilisateur soit inscrit sur le site afin de référencer tout fichier MIDI créé. Les interfaces de connexion et d'inscription sont disponibles sur la page d'accueil ou sur le logo utilisateur en haut à droite de l'écran.



#	Lien	Mélodie	Date	Action
#1	Ecouter la musique	0:00 / 0:19	19/06/2023 16:00	
#2	Ecouter la musique	0:00 / 0:19	19/06/2023 15:57	
#3	Ecouter la musique	0:00 / 0:19	19/06/2023 15:18	



## Usage

OrchestrAI a été pensé pour être simple d'utilisation. La plateforme utilisateur est construite de telle manière à guider l'utilisateur au mieux vers la conversion de fichiers MIDI.

Comme expliqué ci-dessus, les principaux services nécessitent une connexion / inscription de la part de l'utilisateur. Cette étape permettra à l'utilisateur d'accéder aux précédentes conversions réalisées.

Une fois connecté, l'utilisateur peut accéder à la page de conversion pour fournir un lien Youtube et cliquer sur "Démarrer". Afin d'éviter tout conflit lors de la génération du fichier, un temps de chargement apparaît sur l'interface. Le fichier généré, l'interface se débloque et l'utilisateur aura la possibilité d'écouter l'audio, de visualiser les mesures de celui-ci et de le télécharger.

Dans le cas où l'utilisateur souhaite consulter ses précédentes conversions, il trouvera son historique depuis son profil en haut à droite. Il aura la possibilité d'écouter ses précédents audios, de les télécharger ou de les supprimer.

# Architecture technique

## Etat de l'art

Pour créer notre modèle, nous avons recherché plusieurs papiers scientifiques existants qui se basaient sur la prédiction de musique. C'est un domaine qui a été largement recherché et qui permet souvent l'expérimentation de nouveau modèle pour tâche de d'apprentissage supervisée complexe.

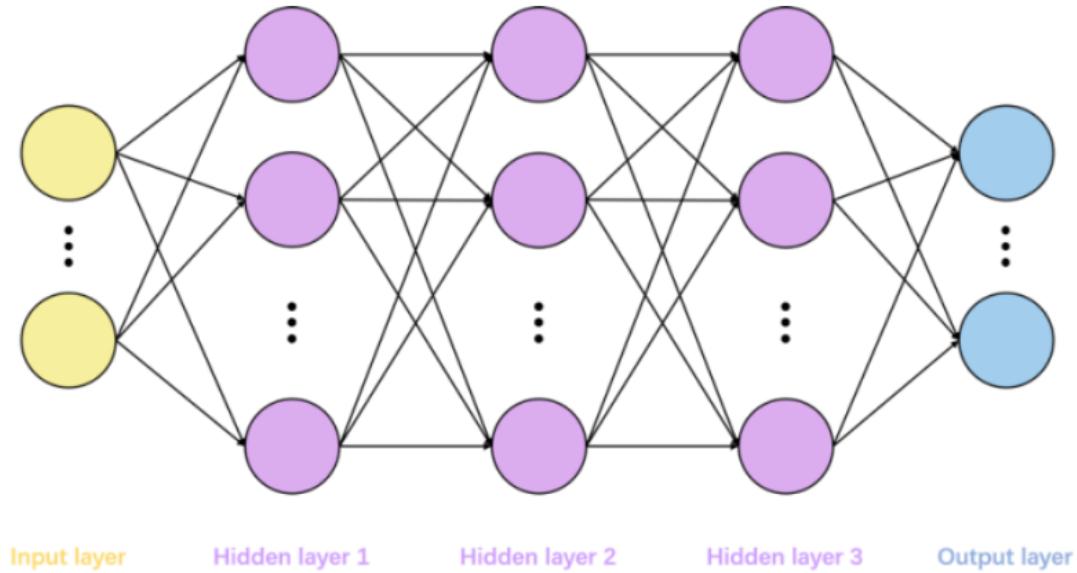
Le point commun que l'on a pu remarquer sont les données d'entrée, en effet la plupart des scientifiques avait plus pour ambition d'améliorer des méthodes existantes en se basant sur un dataset composé de peu de musiques (~150 max) mais où la musique correspondait parfaitement au fichier midi. Le modèle généré n'était pas généraliste et ce n'était pas d'ailleurs son but. Malgré tout, nous nous sommes inspirés de ceux- ci pour modéliser notre modèle.

Pour la construction de nos données d'entrée et d'un de nos modèles nous nous sommes basés sur "Music Transcription Using Deep Learning" [\[1\]](#) par Luo Qi Li, Isabella Ni et Liang Yang de l'université de Stanford.

C'est un projet de recherche qui examine l'application de deux méthodes d'apprentissage profond, DNN (Deep Neural Network) et LSTM (Long Short-Time Memory), pour automatiser la transcription musicale. Les auteurs transforment les fichiers audio en spectrogrammes pour en extraire leurs caractéristiques puis utilisent des méthodes d'apprentissage profond qui ont l'avantage d'apprendre des caractéristiques complexes dans la transcription musicale.

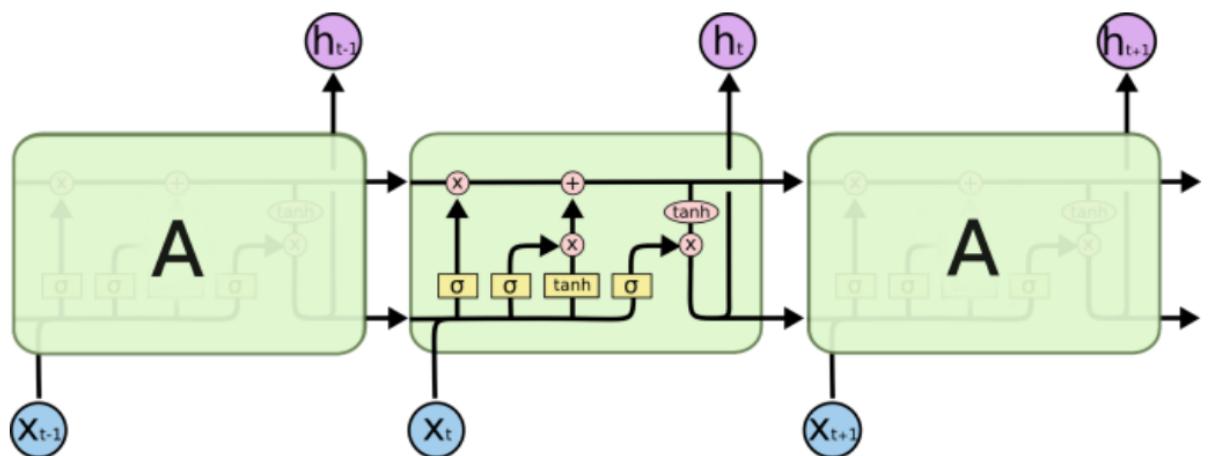
Ce papier a été très important pour nous car il a posé les fondations de notre modèle en faisant une introduction des travaux antérieurs dans ce domaine mais aussi en parlant de leurs transformations sur la donnée et en leur donnant un pseudo code utilisé.

Ils test avec un DNN et du dropout.



*exemple de DNN classique*

Mais aussi en utilisant un LSTM, qui sont des réseaux de neurones. Contrairement aux réseaux de neurones classiques, les LSTM ont des connexions de rétroaction, ce qui leur permet de traiter des séquences complètes de données. Cette caractéristique les rend idéaux pour le traitement et la prédiction des données, tels que la reconnaissance d'écriture manuscrite, la reconnaissance vocale, la traduction automatique, la détection d'activité vocale, le contrôle de robots, les jeux vidéo et les soins de santé. Les LSTM sont utilisés dans de nombreux domaines où il est important de traiter des données séquentielles.



*exemple de LSTM*

Les LSTM sont composés de cellules de trois portes (gates) :

- une input gate qui reçoit les données nouvelles

- une forget gate qui récupère les données de la cellules précédentes mais on choisissant de n'en garder qu'une partie
- une output gate qui envoie les données en sortie

Nous nous sommes aussi aidés du papier scientifique “Reinforcement Learning with Long Short-Term Memory”<sup>[2]</sup> par Bram Bakker, où il mélange du reinforcement learning et LSTM. Selon ce papier, l'apprentissage par renforcement sans modèle : RL-LSTM, utilisant l'apprentissage Advantage et l'exploration dirigée, peut résoudre des tâches non-Markoviennes avec des dépendances à long terme entre les événements pertinents.

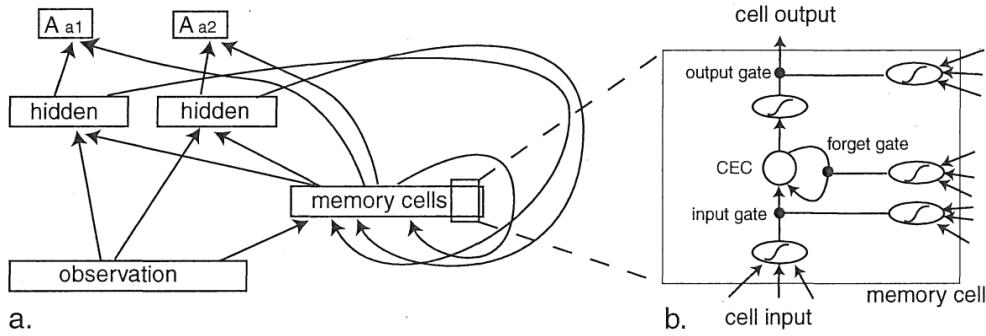


Figure 1: a. The general LSTM architecture used in this paper. Arrows indicate unidirectional, fully connected weights. The network's output units directly code for the Advantages of different actions. b. One memory cell.

*Architecture utilisé dans le papier “Reinforcement Learning with Long Short-Term Memory”*

Nous avons aussi étudié d'autres papiers mais qui nous n'avons pas pu prendre comme sujet d'études par manque de temps ou lier à d'autres contraintes. Malgré tout nous pouvons noter : "Residual Shuffle-Exchange Network for Fast Processing of Long Sequences"<sup>[3]</sup> by Andis Draguns, Emīls Ozolins, Agris Sostaks, Matīss Apinis, Karlis Freivalds qui présente un modèle fiable (60% de réussite) au détriment d'une grande complexité à implémenter.

## Briques technologiques

Dans le cadre de notre projet, nous avons utilisé deux briques technologiques nous permettant de mettre en place l'architecture de OrchestrAI d'un point de vue utilisateur et gestion de l'infrastructure.

- PlanetHoster:

Pour assurer un hébergement web, nous avons choisi la plateforme PlanetHoster. Celui-ci propose un hébergement complet avec un nom de domaine et une base de données incluse pour la gestion utilisateur.

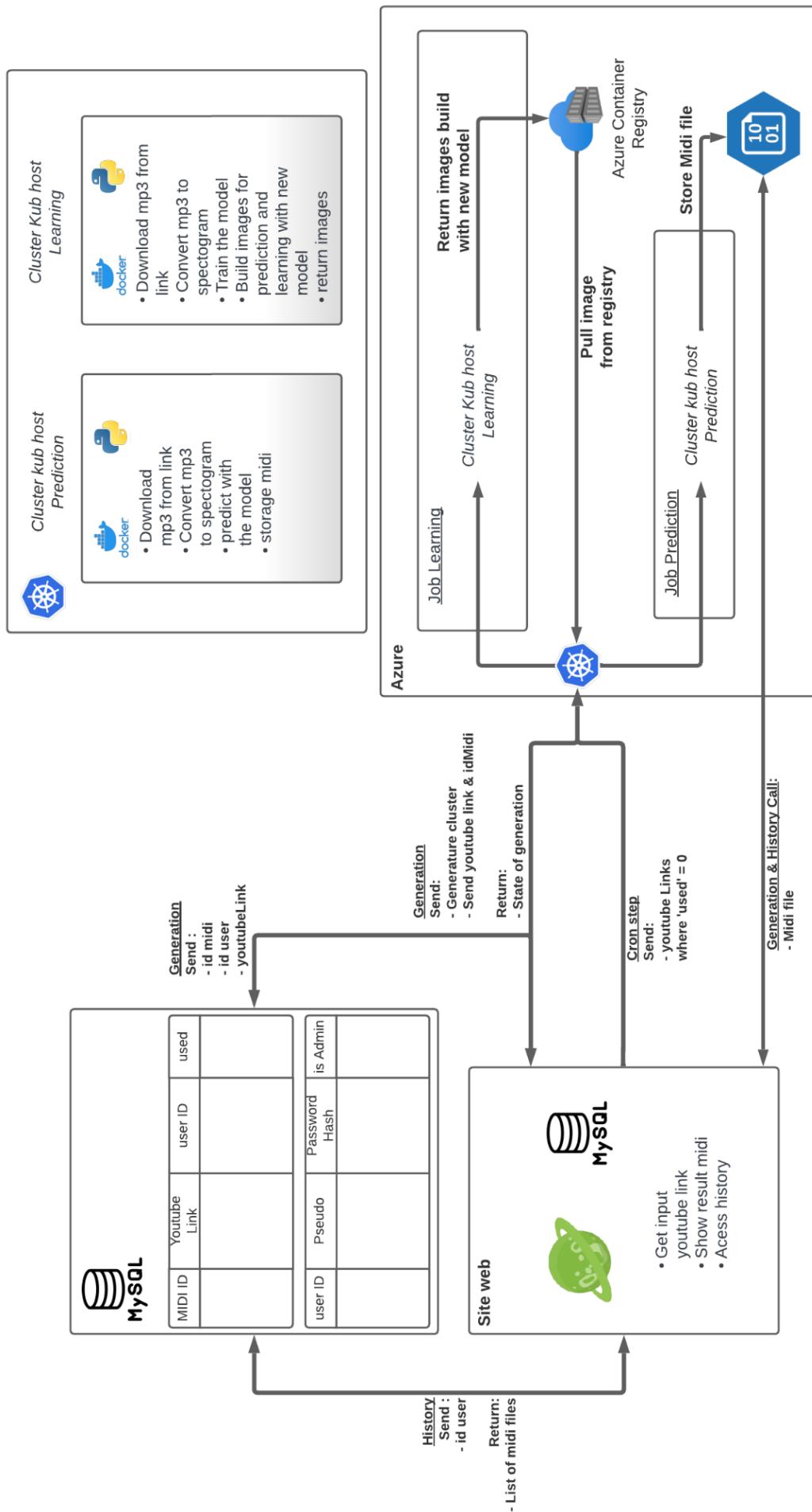
L'hébergeur propose par la même occasion un stockage illimité avec l'abonnement choisi. Cependant, nous utilisons uniquement pour stocker les fichiers nécessaires au fonctionnement du site web.

- Azure:

Pour la génération des fichiers MIDI ainsi que l'apprentissage en continu de notre modèle, nous avons opté pour la plateforme Cloud Azure. Le choix de cette plateforme par rapport à AWS et GCP a été fait sur plusieurs critères:

- ➔ L'interaction entre le langage PHP et la plateforme cloud pour activer les différents services tels que les clusters, les blobs storages, etc.
- ➔ Les faibles coûts pour les services utilisés selon le temps de fonctionnement et la région,
- ➔ La facilité d'utilisation par le groupe acquis lors de différents projets scolaires.

# Schéma de l'architecture du projet



## Choix architecturaux

Afin de proposer une architecture fiable et optimisée, nous avons minimiser l'utilisation des différents outils à notre disposition. Le site n'étant que l'interface utilisateur et ne proposant pas d'autres configurations possibles, l'intégralité de nos choix pour l'amélioration et la fiabilité du projet repose sur Azure. Lors de la construction et la configuration de notre architecture azure, plusieurs problématiques sont apparus et dont nous avons dû trouver une solution :

- Le choix du stockage

Dans l'obligation de stocker les différents fichiers midi générés, il était nécessaire de définir la meilleure méthode de stockage. Celui-ci devait être accessible facilement depuis le script de génération et pouvoir facilement communiquer avec le site.

Nous avons donc opté pour un blob storage avec un stockage à froid pour minimiser les coûts. Les fichiers MIDI étant très légers et le prix du stockage du blob n'augmentant qu'à partir d'un certain palier de stockage, les coûts générés sont minimes.

Nous avons opté pour un stockage azure et non site afin de s'assurer de la bonne génération du fichier et de limiter les connexions. Les fichiers pourront être appelés sur le site pour le téléchargement par la suite.

- Le choix de l'environnement Azure

Afin de générer rapidement un environnement pour la génération du fichier, nous avons décidé d'utiliser le registre d'images d'Azure. Celui-ci nous permet de stocker les différentes images de nos environnements que nous pourrons appeler par la suite.

Un des avantages que propose ce système est la prise en charge des versions d'images. Cette fonctionnalité nous a permis de rapidement mettre à jour nos environnements et nos modèles sans devoir recourir à la suppression de l'image.

- Le service kubernetes

Afin de générer notre fichier midi, nous devions exécuter un script dans un environnement contenant tous les packages nécessaires et accessibles instantanément. Ces conditions obligent à maintenir un environnement constamment ouvert et engendrent des coûts trop élevés. Pour répondre à cette contrainte, nous avons utilisé le service kubernetes d'Azure pouvant créer des environnements à partir d'images stockées dans le registre.

Un des avantages d'AKS est la possibilité de configurer cet environnement sur sa durée de fonctionnement. En créant un environnement "job", celui-ci restera ouvert sur le temps de l'exécution du code. Une fois tournée, le code supprime automatiquement le job.

Cette configuration d'Azure permet de limiter les coûts tout en assurant l'accès permanent au service. Elle assure un accès pour chaque utilisateur grâce à sa possibilité de gérer plusieurs jobs à la fois.

# Données mises en œuvre

Notre première étape a été de choisir la donnée que nous souhaitions utiliser pour l'apprentissage de notre modèle. Nous souhaitions créer un modèle généraliste qui pourrait prendre en entrée un grand nombre de genres de musique différents mais qui rendrait en contrepartie l'apprentissage plus complexe.

## Récupération de la donnée

Pour constituer notre dataset, nous devions récupérer 2 éléments pour chaque musique récupérée: Le fichier MIDI et le fichier MP3 lui étant associé. N'existant pas de bases de données public regroupant ces deux informations, nous avons eu l'obligation de récupérer par nous même la donnée en plusieurs étapes.

- Récupération du fichier MIDI

A ce jour, plusieurs sites mettent à disposition des fichiers MIDI. Dans le cadre de notre projet, nous avons choisi le site proposant le plus grand choix de fichiers que ce soit en nombre ou en diversité : MuseScore.

Accessible grâce à un compte, l'objectif initial était de mettre en place un scraper parcourant l'entièreté du site en cliquant sur chaque lien de fichier et de cliquer sur le bouton "télécharger". Cependant, une partie de ces fichiers considérés comme "pro" n'étaient pas accessibles sans un compte premium.

Pour contourner ce soucis, nous avons dû utiliser une application fonctionnant par CLI dont la mission était de récupérer et télécharger un fichier MIDI depuis un lien provenant de Musescore: LibreScore. Fonctionnant pour la plupart des fichiers, nous avons pu limiter la perte imposée par le compte premium.

Le nouvel objectif pour la récupération des fichiers MIDI était donc de récupérer les liens et titres des partitions sur MuseScore et de récupérer le fichier via l'application LibreScore. Ce processus fût plus rapide que la méthode initiale car nous n'avons plus la contrainte de charger chaque page de partitions sur MuseScore pour cliquer sur le bouton "Télécharger".

- Récupération du fichier MP3

Pour récupérer le fichier MP3 associé au fichier MIDI, il n'y avait que très peu de possibilités à notre disposition. Nous avons donc choisi la solution ayant un large choix de fichiers MP3 disponible gratuitement et rapidement: Youtube.

Toujours avec un système de scraper, l'objectif était de trouver un lien Youtube à partir du nom du fichier MIDI et de le convertir en fichier MP3. Plusieurs filtres ont été appliquées pour améliorer la recherche comme:

- Définir une plage de maximum pour l'audio. les partitions de musiques récupérées n'étant que des partitions de musiques existantes ou des extraits de concerts, le temps ne pouvait pas dépasser 1 heure.
- Retirer tous les contenus "shorts". Youtube a mis en place des vidéos "shorts", des vidéos courtes où les utilisateurs utilisent des extraits de musique et non son entièrement.
- Ajouter des tags pour améliorer la recherche. Pour éviter les clips vidéos de chaque musique ayant très souvent une introduction sans musique, nous avons ajouté le tag "audio".

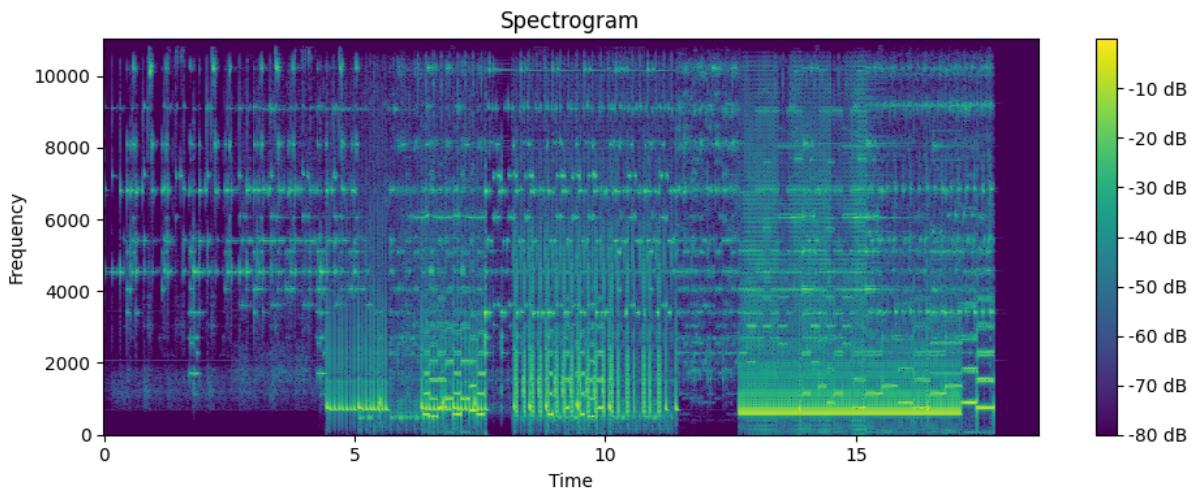
A partir de ce lien Youtube, nous avons utilisé divers packages python ayant la capacité de récupérer le contenu MP4 de la musique et de le convertir en contenu MP3. Comme pour les fichiers MIDI, les packages ont fonctionné pour la plupart des liens et nous ont fourni une base de données assez complète pour la réalisation de notre projet.

Malgré le retrait d'environ 5 000 fichiers MP3 et de fichiers MIDI dont l'audio ou la partition n'a pas pu être téléchargée, le nombre total de partitions récupérées s'élève à environ 31 000.

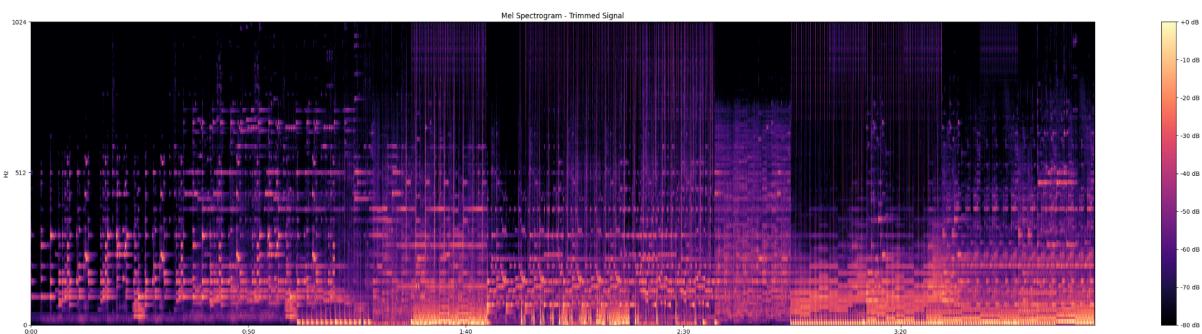
## Traitement de la donnée

Une fois les fichiers MP3 obtenues nous avons dû les transformer pour les faire ingérer à notre modèle. Nous avons regardé dans plusieurs papiers scientifiques pour savoir comment les chercheurs procédaient, et avons dû créer des spectrogrammes. Il y a plusieurs façons de générer les spectrogrammes à partir des MP3 et nous avons essayé de viser ce qu'il semblait le plus visible avec nos données (multi instrument et voix).

En se basant sur les papiers scientifiques, une simple transformé de fourier suffisait mais nous trouvions que cela n'était pas lisible

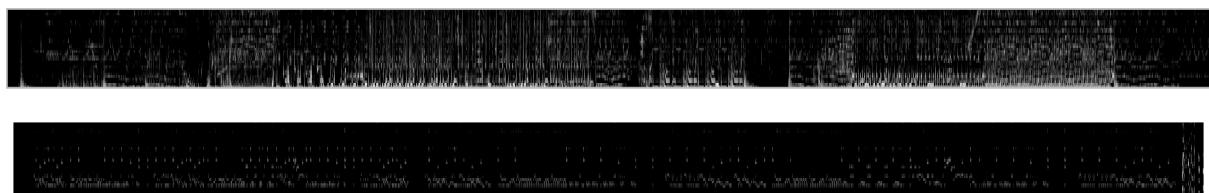


Donc nous avons regardé d'autre façon de représenter la donnée et nous avons opté pour des mel spectrogrammes. La principale différence est que les mel-scaled spectrogrammes permettent de mieux représenter les basses qui peuvent dans notre cas représenter la mélodie que l'on cherchons à avoir.



Nous avons par ensuite décidé de filtrer ces spectrogrammes pour limiter l'information et la rendre plus lisible.

Tout d'abords en noir et blanc, puis en filtrant les fréquences les plus basses (qui représente particulièrement les percussions) car celà générat un gros "bruit"



Nous avons aussi uniformisé nos spectrogrammes pour pouvoir les faire ingérer à notre modèle. Pour cela, nous avons fait le choix de mettre à la même taille chacun de nos spectrogrammes quelque soit leur longueurs.

La dernière étape de la transformation a été de réduire la taille globale de la représentation de nos spectrogrammes pour ensuite le découper dans un numpy array qui prend comme shape (100, 1600, 1).

Les fichiers MIDI possèdent de nombreux paramètres par note ( le pitch, le temps de début et de fin de la note, vitesse, canal, tempo et bien plus). Lors de l'apprentissage nous avons décidé de ne sortir que trois valeurs par note : le pitch, le temps de début et de fin de note. Nous avons fixé les autres paramètres de manière fixe et ces mêmes paramètres seront utilisés durant les prédictions.

Nous avons aussi décidé de limiter le nombre de notes générées à d'abords à 1 667 puis à 834 pour des raisons de facilité de traitement et de réduction de mémoire nécessaires pour l'utilisation de modèle plus puissant.

## Evolution de la donnée

Pour continuer à améliorer notre modèle et les fichiers générés, nous utilisons les données Youtube fournies par l'utilisateur. Grâce à un cron, un script va récupérer tous les fichiers MP3 liés aux liens Youtube.

Souhaitant une partition non générée pour l'amélioration du modèle dans un premier temps, nous effectuons une requête Google avec le nom de l'audio récupérée ainsi que le tag "MuseScore". Grâce au lien MuseScore fourni par la requête Google, nous utilisons de nouveau LibreScore pour récupérer et télécharger le fichier MIDI.

Une fois le modèle mis à jour avec les différents liens Youtube, ceux-ci sont référencés comme utilisés dans la base de données du site pour éviter un entraînement sur les mêmes données.

# Algorithmes mis en œuvre

## Algorithme de scrap et de traitement

Pour mener à bien la récupération des différentes données, plusieurs librairies ont été utilisées. En fonction de l'étape de récupération, celles-ci ont plus ou moins été sollicitées et ont influencé l'évolution du script selon leurs avantages et leurs contraintes.

- Selenium

Imitant le comportement humain, Selenium nous a permis de récupérer le contenu des pages de MuseScore pour les liens. Un des avantages que propose Selenium est que nous ne sommes pas restreint à la page grâce à son interaction avec les zones de texte et les boutons.

- BeautifulSoup

Ayant la même fonction principale que Selenium, BeautifulSoup n'est cependant pas capable d'interagir avec le contenu de la page récupérée. En fonction de notre besoin, nous avons privilégié l'utilisation de BeautifulSoup par rapport à Selenium nécessitant moins de ressources.

- yt-dlp et PyTube

Pour notre besoin en fichiers MP3, nous avons utilisé plusieurs librairies ayant la même fonction: récupérer le contenu audio sur Youtube dans un format MP3 ou MP4. Suite à de nombreux problèmes de compatibilités et de mises à jour avec Pytube, nous avons dû utiliser yt-dlp, ayant le même principe mais avec une conversion en fichier MP3 et non MP4 comme PyTube.

- Pretty MIDI

Grâce à cette librairie, nous avons pu facilement extraire des informations à partir des fichiers MIDI, telles que les notes, les accords, les instruments etc... Celle-ci a été utile lors de la génération de nos fichiers MIDI pour définir l'emplacement des notes ou lors du traitement des fichiers récupérés pour l'apprentissage.

## Algorithme d'entraînement

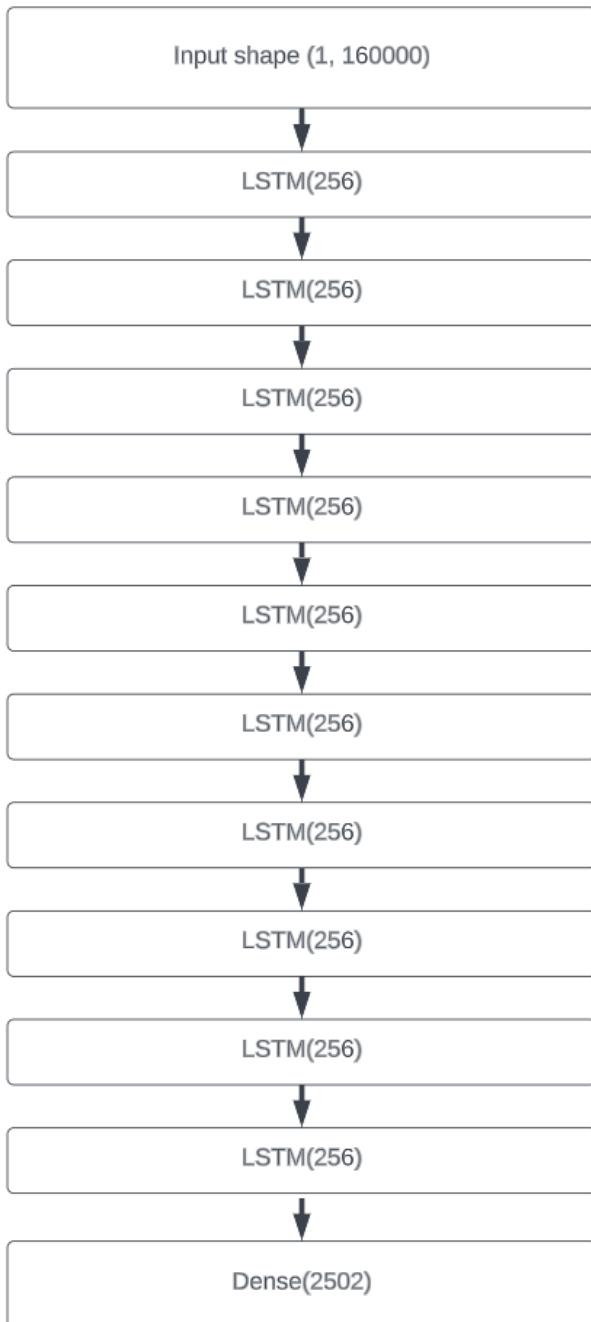
Au niveau du modèle, nous nous sommes servis de la librairie Keras pour le créer et l'entraîner. À l'aide de couches Dense et LSTM, nous avons essayé de reproduire différentes architectures que nous avons pu observer dans divers papiers scientifiques, notamment des RNN.

En ce qui concerne l'entraînement, nous avons beaucoup fait varier le nombre de couches ainsi que le nombre de neurones, mais aussi le nombre d'epochs, étant un facteur important dans l'apprentissage d'un modèle. Pour le calcul du loss, nous avons utilisé la binary cross entropy et le mean square error.

Ce que nous avons pu constater quant aux résultats, c'est que nous ne parvenions pas à réduire significativement le loss, compris entre 100 000 et 2 000 000, les fichiers MIDI obtenus ne sont pas exploitables en tant que partition.

Steps per epoch : 10  
epochs : 100  
train on ~30000 musics

loss  
Categorical\_crossentropy

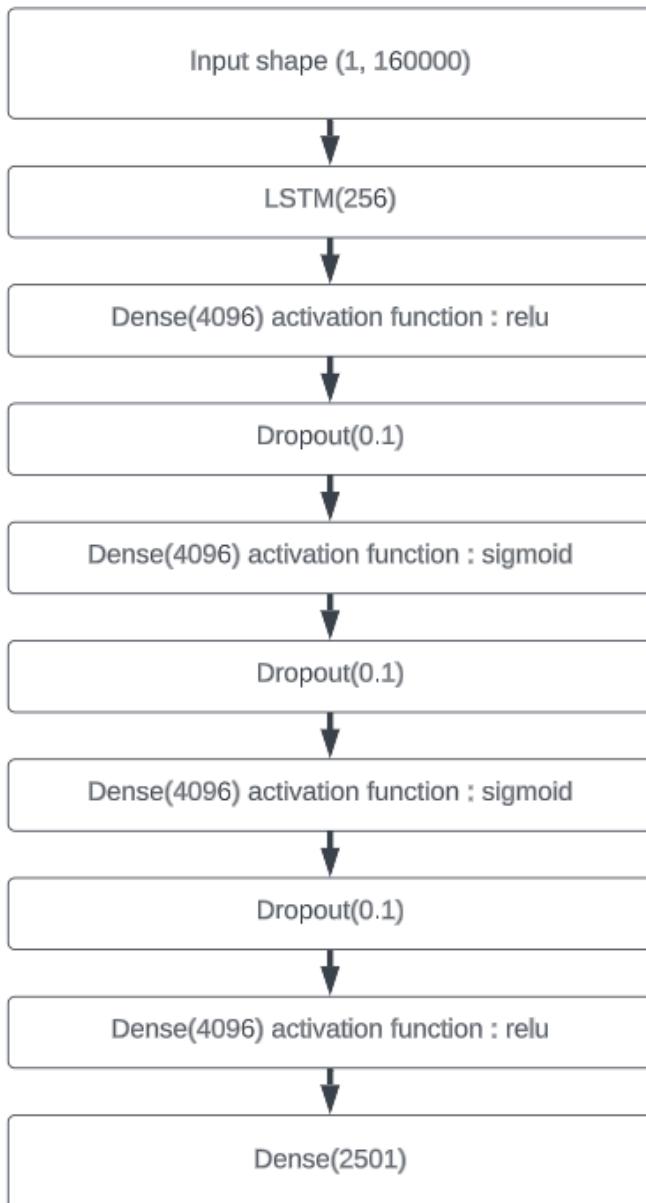


Le premier modèle que nous avons mis en place et que nous avons entraîné est un DNN de couches LSTM. Nous avions essayé d'obtenir des résultats convenables en faisant varier le nombre de couches et le nombre de neurones de chaque couche, sans succès.

En sortie, nous récupérions 2502 nombres : 834 notes, 834 timing de départ et 834 timing de fin. Cette valeur a été définie arbitrairement.

Steps per epoch : 10  
epochs : 3000  
train on ~30000 musics

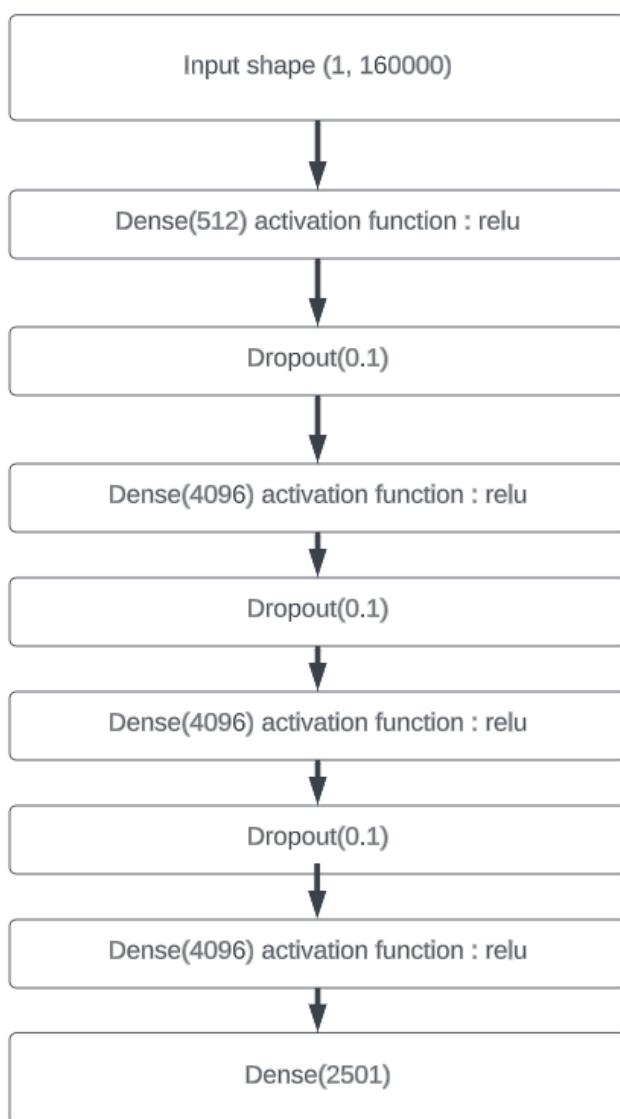
loss  
Categorical\_crossentropy



Pour ce modèle-ci, nous avons tenté de faire un mix de ce que nous avons pu trouver dans des papiers scientifiques. Nous avons donc essayé de mettre une couche de LSTM en entrée puis de la faire suivre par un DNN. Nous avons fait varier le nombre de couches Dense, les fonctions d'activation et les nombres de neurones. Les résultats observés étaient un peu meilleurs que pour le modèle précédent.

Steps per epoch : 10  
epochs : 500  
train on ~30000 musics

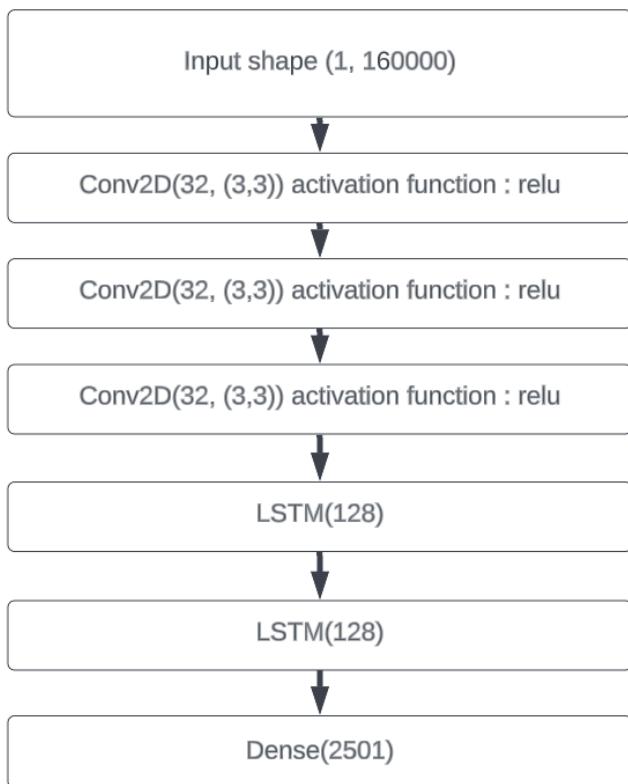
loss  
Categorical\_crossentropy



Dans ce DNN, la première couche est un peu plus petite que les autres car la multiplication de 160000 par son nombre de neurones dépasse vite la mémoire de nos machines. Nous avons fait varier le nombre de couches, le nombre de neurones et les fonctions d'activation. Les résultats observés n'étaient pas meilleurs que pour le premier modèle.

Steps per epoch : 10  
epochs : 200  
train on ~30000 musics

loss  
Categorical\_crossentropy



Nous avons aussi rapidement essayé de faire un modèle mélangeant Conv2D et LSTM.  
En faisant varier les paramètres des Conv2D et des LSTM, nous n'avons pas obtenu de résultats probants ici.

## Job kubernetes

Pour appeler la création de job Kubernetes depuis PHP, nous avons utilisé la librairie PhpK8s qui n'est pas officielle car Microsoft n'en possède pas. Elle est malgré tout reconnue par l'entreprise et est proposée sur leur page de documentation.

Au sein de nos jobs Kubernetes, nous avons utilisé de nombreuses librairies :

- yt-dlp : permet de télécharger des audios youtube sans avoir besoin de se connecter
- azure-storage-blob : permet de se connecter et de lire / écrire des fichiers dans un blob azure
- beautifulsoup : permet de récupérer les informations d'une page web
- requests : permet de charger une page web
- azure-cli : permet d'utiliser Azure CLI pour pouvoir build sur Azure Image Registry

Ainsi qu'une majeure partie des librairies utilisées précédemment pour la récupération de données.

## Gestion de projet

### Planning de réalisation

L'idée de ce projet a émergé très tôt dans notre groupe, nous avons donc commencé par récupérer un maximum de données (MP3 et fichiers MIDI) lors du premier semestre, pendant environ 3 mois.

Nous avons ensuite passé 1 mois à formater les données MP3 pour qu'ils puissent être ingérés par un modèle de deep learning, il a donc fallu les convertir en spectrogramme, puis en numpy arrays.

Nous avons fait de même avec les fichiers MIDI (conversion MIDI -> numpy arrays) pendant une semaine.

Puis nous nous sommes concentrés sur la création et l'entraînement du modèle qui nous ont pris 3 mois.

Enfin nous avons monté notre architecture Azure, créé notre site web et géré toute la partie utilisateurs en 2 mois.

## Répartition du travail

Nous avons principalement travaillé en pair programming, cela dit, Alexandre a principalement pris en main le scrapping, le site web et l'architecture Azure, Arthur a principalement travaillé sur le formatage des données, le modèle d'apprentissage et l'architecture Azure et Steven a principalement travaillé sur le formatage des données et le modèle d'apprentissage.

## Problèmes rencontrés

Lors de la réalisation de notre projet, nous avons rencontré divers problèmes lors de chaque étape de celle-ci.

Le premier problème s'est produit lorsque nous devions rassembler des données pour notre dataset, il nous fallait une source de fichiers MIDI fiable, gratuite et pouvant en fournir en masse. Nous avons trouvé le site Musescore, qui permet de récupérer des fichiers MIDI traduits par des utilisateurs du monde entier. Nous avons donc tout simplement scrappé ces fichiers avec la musique correspondante afin de former notre dataset.

Nous avons ensuite rencontré un souci avec le spectrogramme des MP3. Au début, nous pensions qu'il fallait générer le spectrogramme de la musique, puis enregistrer celui-ci au format image pour finalement le récupérer afin de le convertir en tableau numpy. Évidemment, le temps de traitement était long et n'était pas acceptable pour un dataset de plus de 30 000 musiques. Cela dit, nous avons trouvé un moyen de faire la même chose sans passer par la conversion en image, grâce à la librairie matplotlib.

Le troisième problème rencontré concernait le formatage du dataset : nous étions partis du principe que nous devions faire des sorties dynamiques au niveau du modèle, puisque les musiques ont des durées différentes, et que leur fichier MIDI associé avait une taille variable également. Comme solution, nous avons décidé de tronquer arbitrairement les musiques et les fichiers MIDI pour que toutes les sorties aient la même taille.

Nous avons aussi eu une palette de problèmes au niveau de notre modèle, concernant les couches par exemple, nous avions eu des difficultés à trouver les bonnes shapes d'entrée et de sortie.

Ayant un dataset de plus de 30 000 exemples, nos machines ne pouvaient supporter de charger l'ensemble dans la mémoire, nous avons donc dû passer par un générateur qui permet de charger des batchs d'une taille définie.

Nous avons aussi essayé d'ajouter du Reinforcement Learning au modèle en intégrant un système de reward, cependant ce dit système nécessitait d'avoir un loss personnalisé, ce dernier étant un peu difficile à harmoniser avec le générateur précédemment ajouté. Nous avons finalement réussi, mais en approfondissant nos recherches, nous avons constaté que ça n'était pas suffisant pour faire du Reinforcement Learning et en constatant la réelle complexité de l'ajout, nous avons finalement abandonné l'idée, l'investissement estimé étant trop lourd.

Le seul problème que nous ne sommes pas parvenus à résoudre pour des raisons de moyens se trouve dans la correspondance MP3 / MIDI, en effet, nous savons que les musiques ont des introductions, des refrains et des conclusions qui n'apparaissent pas forcément dans les fichiers MIDI (pour les refrains par exemple, il sont souvent joués plusieurs fois dans la musique, mais dans le fichier MIDI il n'apparaîtra peut-être qu'une seule fois).

Dû à des accès limités sur les comptes gratuits d'Azure, les fonctionnalités disponibles dans les machines virtuelles étaient limitées au niveau de Kubernetes. De même, avoir un compte gratuit a complexifié la création et la gestion de certains services au sein d'Azure et demandait parfois quelques pirouettes techniques pour arriver à nos fins.

Aussi, lors du choix de ces clusters, nous avions la possibilité de choisir entre plusieurs zones et régions. Malheureusement, les différentes machines disponibles peuvent grandement différer en termes de puissance mais aussi de prix selon les régions. En voulant garder bonne conscience, nous avons décidé de prendre des serveurs ouest-européens (énergie générée principalement par du nucléaire), nous nous sommes rendus compte que le prix augmentait de presque 50% pour des performances similaires par rapport aux zones

US ou nord-européen. C'est pourquoi, pour le bien de notre budget, nous avons décidé, à regret, d'utiliser la région nord-européenne.

Notre interface front a été développée en PHP, nous avons voulu optimiser la création des jobs Kubernetes directement depuis un bouton fait en PHP. Nous nous sommes rendus compte que Microsoft n'a pas de package PHP pour se connecter avec Azure et avons dû utiliser un package reconnu par Microsoft mais non officiel qui peut donc générer un point de failure pour la maintenance dans le temps.

Nous avons rencontré un problème également avec l'auto building entre Docker et Azure, l'idée était de lancer un Job qui permette de faire l'auto-apprentissage du modèle, cependant en installant Docker sur la machine virtuelle nous avons rencontrés divers problèmes de compatibilités (Kubernetes n'utilisant pas Docker).

Nous avons donc décidé de simplement uploader notre modèle sur Azure Registry, néanmoins il n'est pas possible de faire un apt get install sur Azure CLI car notre image python utilise Debian 12 qui n'est pas compatible aujourd'hui mais qui sera dans une future proche (mi-2023).

# Résultats obtenus

## Bilan fonctionnel

Finalement, notre projet a abouti à une architecture cloud fonctionnelle et automatique, un site web utilisateur qui permet de s'inscrire, de faire des conversions et de garder en mémoire les conversions qui ont été faites précédemment. Notre modèle permet bien de faire une conversion de MP3 à MIDI, même si le fichier MIDI généré ne permet pas de reconnaître la mélodie comme nous l'avions espéré.

Nous avons réussi, avec un budget moindre, à réaliser ce projet pavé d'embûches et à découvrir les différentes technologies utilisées dans le domaine de la génération et de la transcription de musiques. Les coûts sont plutôt bien optimisés pour nos moyens et notre architecture offre la possibilité de faire des traitements à plus grande échelle facilement en ne modifiant que très peu de choses.

## Améliorations possibles

Le principal axe d'amélioration serait de modifier notre dataset pour que les musiques au format MP3 correspondent parfaitement aux fichiers MIDI. La solution la plus simple serait d'engager des personnes spécialisées pour faire ce travail de transcription afin d'avoir un dataset assez précis pour l'entraînement. Si un tel dataset nous était accessible, il nous suffirait d'entraîner un modèle assez performant pour que notre projet soit complet.

Ceci dit, à notre échelle, il serait judicieux de revoir notre façon de standardiser et de normaliser nos données. De plus, on pourrait viser moins large en partant sur un modèle moins généraliste qui se baserait sur le MP3 des fichiers MIDI. Ainsi on aurait une transcription propre mais qui aurait du mal à fonctionner sur des musiques composées de plusieurs instruments, de chants et aurait plus de mal à l'écoute du mixage sonore.

## Sources

[1] : “Music Transcription Using Deep Learning” par Luoqi Li, Isabella Ni, Liang Yang  
<https://cs229.stanford.edu/proj2017/final-reports/5242716.pdf>

[2]: “Reinforcement Learning with Long Term Short Memory” by Bram Bakker  
<https://proceedings.neurips.cc/paper/2001/file/a38b16173474ba8b1a95bcbe30d3b8a5-Paper.pdf>

[3] : “Residual Shuffle-Exchange Network for Fast Processing of Long Sequences” by Andis Draguns, Emils Ozolins, Agris Sostaks, Matiss Apinis, Karlis Freivalds  
<https://arxiv.org/pdf/2004.04662.pdf>