CENTRALE NANTES

Option professionnelle Science et Musique

Rapport de projet

Mélange de mélodies avec MusicVAE

Elève
Olivier Cornet

Professeur encadrant
Mathieu Lagrange

Sommaire

I. Introduction

A. Présentation de Google Magenta

Google Magenta est un projet de recherche open source qui explore le rôle du machine learning en tant qu'outil pour la création musicale[1] développé par Google. Il est composé de nombreux modules, ayant tous leur utilité propre tels que NSynth, GANSynth, MusicVAE, DDSP Timbre Transformer qui permettent d'expérimenter sur l'audio, dans le but de permettre aux artistes d'explorer et de manipuler facilement des éléments de musique.

Dans ce projet, on se penche sur MusicVAE, un auto-encodeur variationnel hiérarchique récurrent pour apprendre des espaces latents de musiques.[2]

B. Fonctionnement de MusicVAE

Pour mélanger des mélodies (melody mixing), on va tenter d'interpoler deux mélodies dans un espace latent, i.e. une base de représentation parcimonieuse des mélodies, qui représente le plus d'information avec le moins de dimensions possible.

Le melody mixing de MusicVAE repose sur l'apprentissage d'un espace latent dont la qualité est mesurée avec 3 attributs:

- L'expression : est-ce que n'importe quel exemple réel peut être traduit dans cet espace latent, sans perdre trop d'information ?
- Le réalisme : est-ce que n'importe quel point de cet espace latent se traduit par un exemple réaliste ?
- La continuité (smoothness) : est-ce que deux points proches de l'espace latent sont similaires ?

Pour apprendre un espace latent, on utilise un **auto-encodeur** qui va encoder les exemples de mélodies dans un espace parcimonieux, de sorte à réduire la dimension du problème, en minimisant la perte d'information. Cependant, cet auto-encodeur ne crée pas des espaces latents *réalistes,* car certains points de l'espace ne représentent aucun exemple réaliste.

Pour régler ce problème, on introduit le auto-encodeur *variationnel*, le **VAE**, qui introduit un variational loss, qui encourage l'auto-encodeur à générer un espace latent avec une certaine structure qui assure que le décodeur produit quelque chose de réaliste.[2]

Par la suite, une fois l'espace latent appris, on peut sélectionner deux morceaux de mélodies (en MIDI) et expérimenter avec, en interpolant les mélodies.

II. Travail réalisé

A. Installation

La majeure partie du travail de ce projet reste l'installation du setup nécessaire pour se servir de Magenta. La démarche initiale est d'abord d'installer Python et Anaconda.

Ensuite, pour installer la version CPU standard de magenta, il suffit d'exécuter *pip install magenta* avec Anaconda. Pip va installer tous les modules nécessaires pour le bon fonctionnement de magenta.

En revanche, si l'on souhaite installer la version GPU, qui permet de faire tourner les calculs sur la carte graphique, ce qui accélère énormément l'apprentissage, la manoeuvre est plus complexe: il faut d'abord mettre à jour le pilote de la carte graphique, puis se référer à un répertoire de wheels d'installation de tensorflow.[3]

Il faut choisir la version de tensorflow souhaitée (pour magenta, on utilisera tensorflow 1.15 car tensorflow 2.0 n'est pas encore supporté) et installer les versions correspondantes de CUDA et cuDNN disponibles sur le site de nVidia. Ensuite, installer le wheel du tensorflow choisi.
Il est loisible à ce point de vérifier si tensorflow reconnaît la carte graphique. Si c'est le cas, on peut installer magenta-gpu avec *pip install magenta-gpu.*

J'ai personnellement réussi à faire fonctionner tensorflow GPU, mais pas Magenta GPU. L'aide sur internet est extrêmement réduite, et, en absence de piste de progression, j'ai préféré réinstaller la version CPU de Magenta pour pouvoir commencer les expérimentations sur MusicVAE.

B. Expérimentations

Le module MusicVAE se présente comme un ensemble de fonctions permettant de construire des modèles d'auto-encodeurs pour interpoler des mélodies, voire même des musiques composées de plusieurs pistes (drums, mélodie, basse).
Il est possible d'entraîner ses propres modèles avec une base de données de mélodies MIDI appropriée. Dans mon cas, j'ai simplement téléchargé un modèle pré-entraîné sur le serveur de Google, car je ne suis pas parvenu à trouver une base de mélodies adaptée à l'entraînement, et le temps d'apprentissage serait de toutes

manières trop long à cause du fait que je n'ai pas pu faire fonctionner Magenta en GPU.

Dans un premier temps, j'ai chargé un modèle pré-entraîné de mélodies pures à 2 temps (32 notes possibles dans le temps). On peut extraire au hasard des mélodies de l'espace latent du modèle et les écouter :

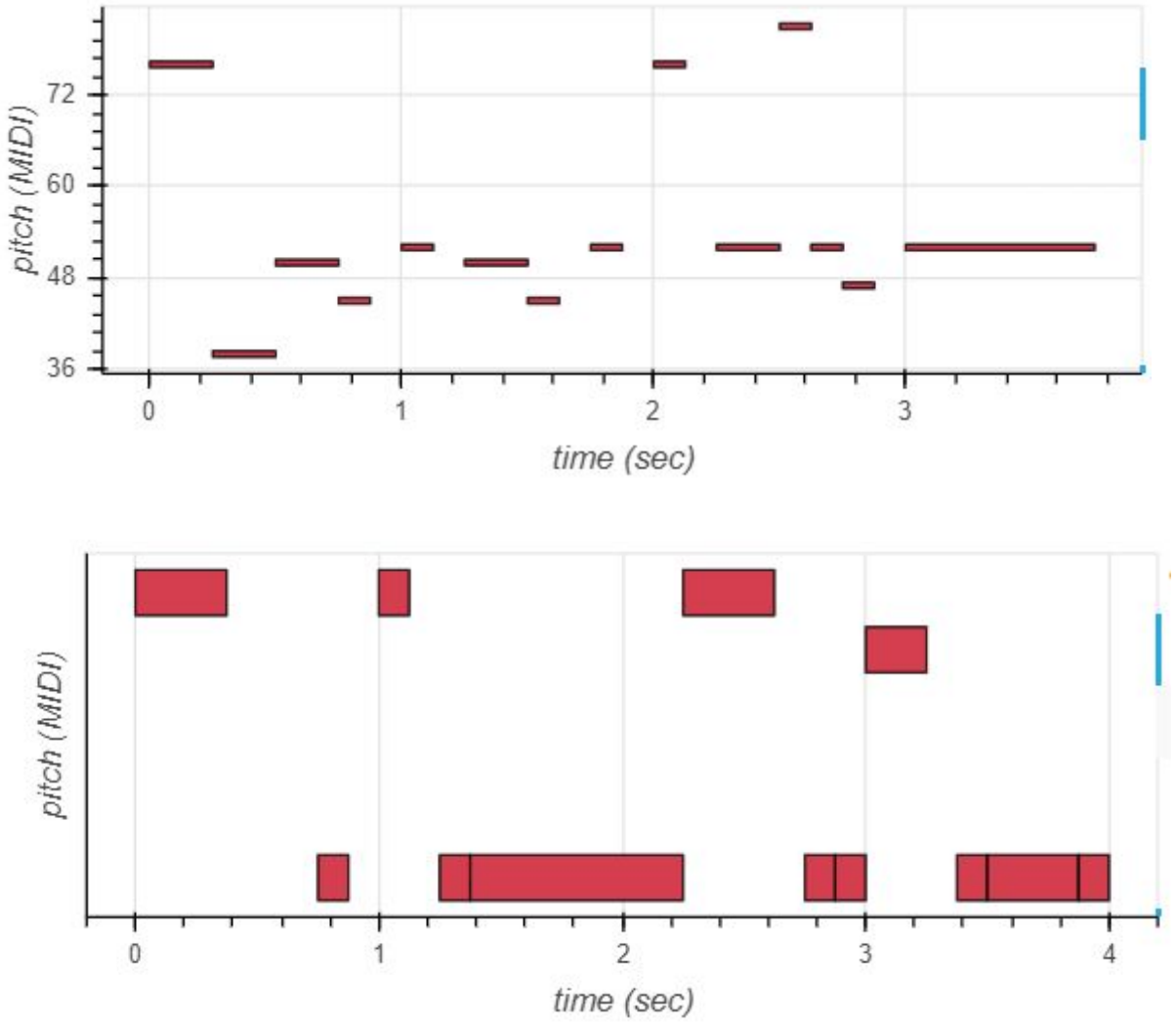

Figure 1 - Deux mélodies aléatoires extraites de l'espace latent du modèle pré-entraîné (son1a.wav et son1b.wav)

En écoutant ces mélodies, on peut valider le réalisme du modèle, car, bien que n'étant pas forcément plaisantes à écouter, ces mélodies sont effectivement réalistes.

Maintenant, pour vérifier la *smoothness* de l'espace latent, on peut effectuer l'interpolation entre ces deux morceaux :

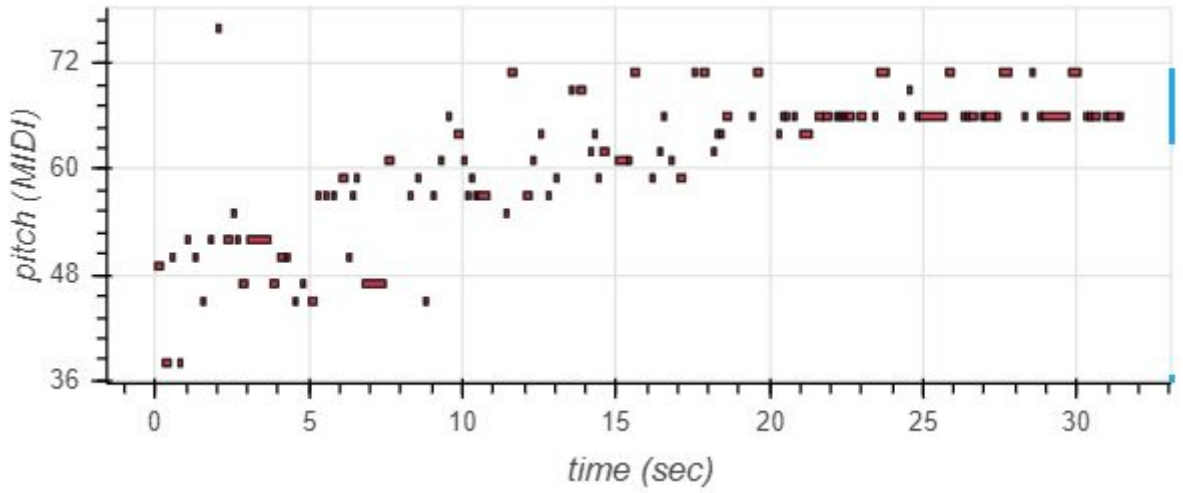

Figure 2 - Interpolation des deux mélodies (température = 0.5)

On interpole les deux mélodies en 8 étapes. On peut reconnaître la première mélodie au début de la piste, et la deuxième à la fin, et la transition entre les deux est continue (*smooth).* Il est possible de modifier le nombre d'étape, ainsi que la température de la fonction softmax: par exemple, avec une température de 1 (précédemment 0.5):

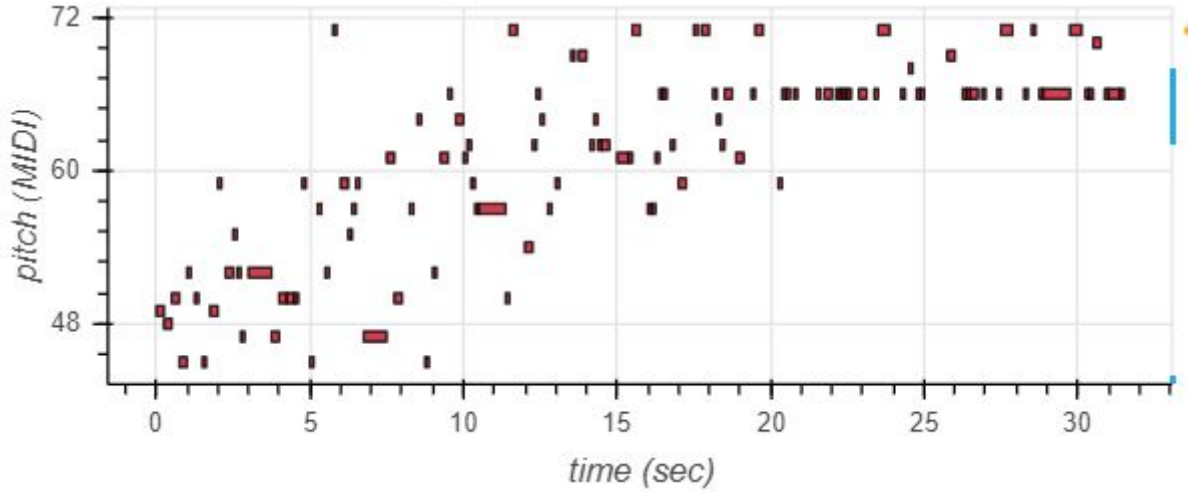

Figure 3 - Interpolation des deux mélodies (température = 1) (son3.wav)

La température augmente la dispersion des notes dans les hauteurs. On pourrait dire que le modèle est plus créatif avec une température élevée.

Test sur des mélodies réelles

On sait que le modèle fonctionne avec des mélodies extraites de son espace latent; il est réaliste et continu. Vérifions maintenant s'il est expressif, c'est à dire qu'on puisse traduire des exemples réels dans l'espace latent.

Observons donc l'interpolation entre deux mélodies réelles:

Ci-dessous, j'ai retranscrit les mélodies de *Radioactivity* de Kraftwerk et de *Around the World* de Daft Punk:

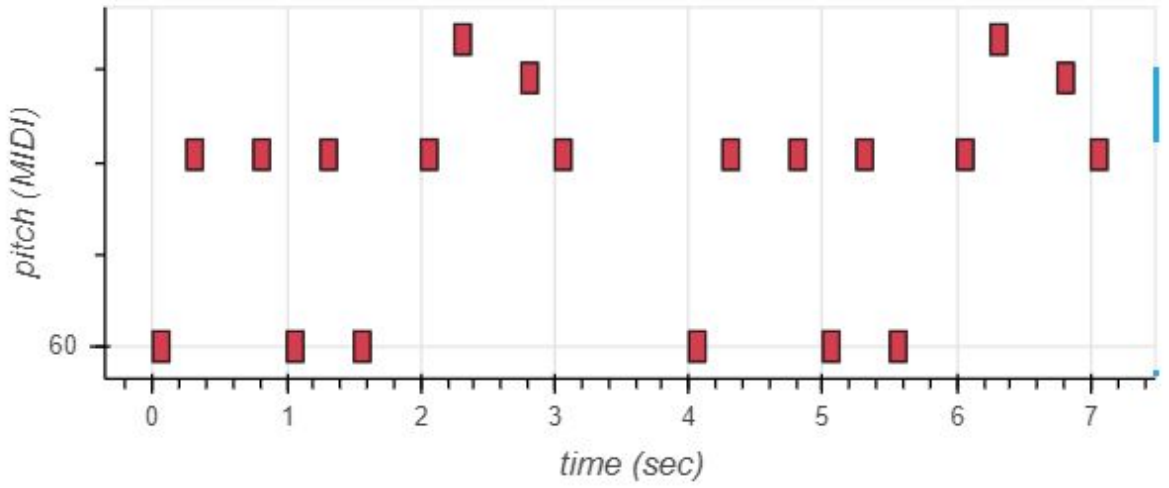

Figure 4 - Radioactivity *de Kraftwerk (son4.wav)*

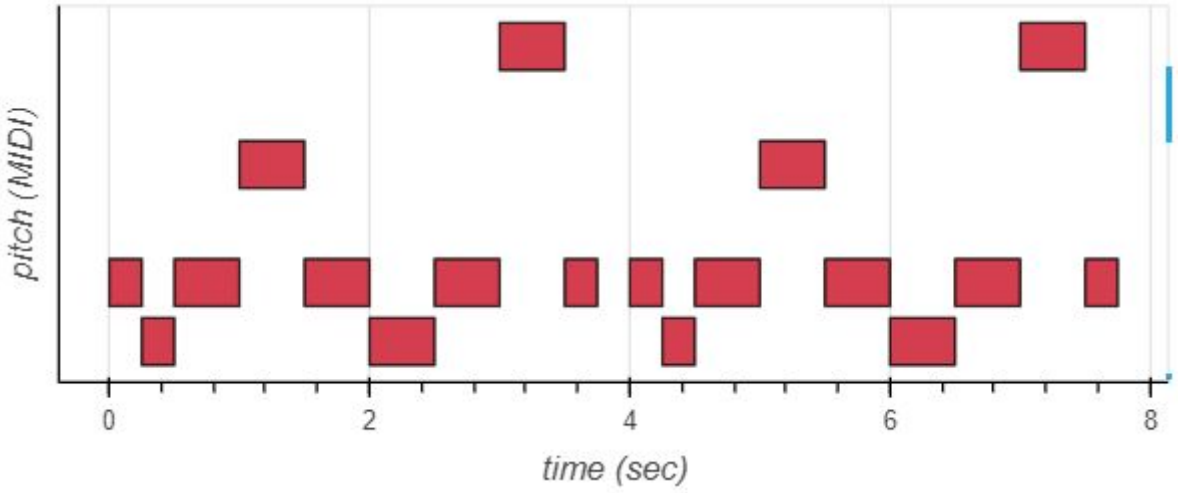

Figure 5 - Around the World *de Daft Punk (son5.wav)*

Lorsqu'on va interpoler ces deux mélodies réelles, le modèle va d'abord les encoder dans son espace latent, les interpoler, et enfin les décoder en un résultat audible:

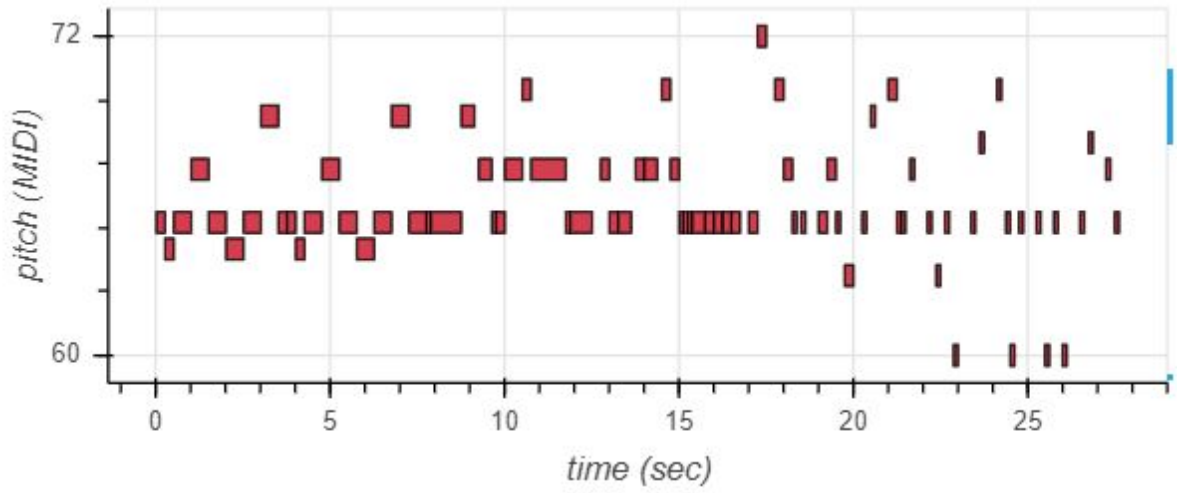

Figure 6 - Interpolation entre Around the World *et* Radioactivity (son6.wav)

Ci-dessus, on transitionne entre Daft Punk à gauche et Kraftwerk à droite. La transition entre les mélodies est encore continue et réaliste. On peut remarquer l'interpolation entre la hauteur des notes mais également leur durée (notes longues dans Around The World et courtes dans Radioactivity)

Vérifions également si l'encodage et l'interpolation fonctionnent si l'une des mélodies est modifiée en hauteur:

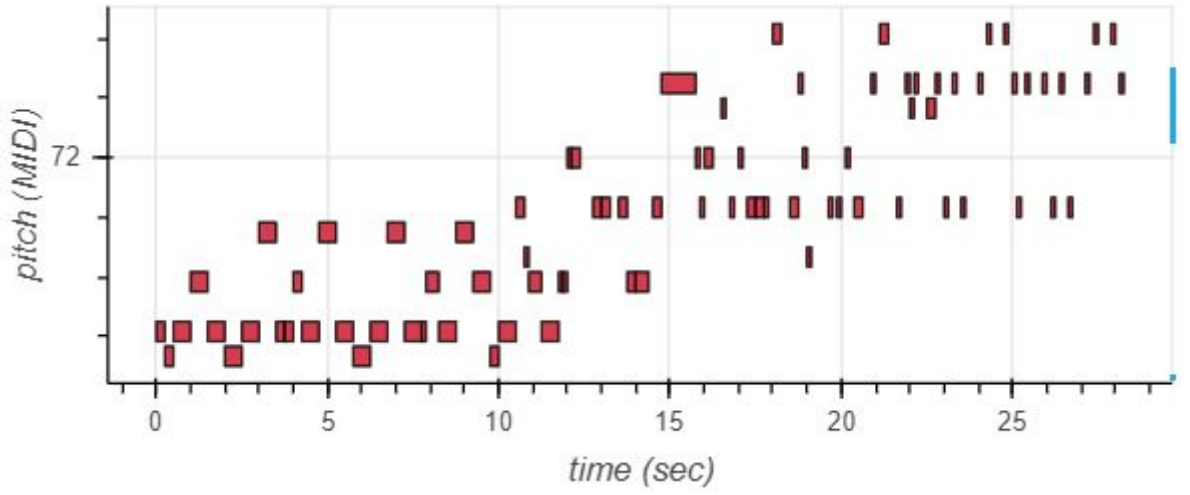

Figure 7 - Interpolation entre Around the World *et* Radioactivity *shifté de 10 demi-tons vers les aigus (son7.wav)*

On peut encore observer une transition smooth, malgré le changement de hauteur.

L'encodage des mélodies dans le modèle est un succès, on peut alors qualifier le modèle d'expressif. Il remplit donc les trois qualités d'un modèle VAE.

Il reste à noter toutefois que certaines mélodies ne peuvent simplement pas être encodées dans l'espace latent. C'est le cas par exemple d'une mélodie entièrement silencieuse, ou composée de trop peu de notes; une exception est renvoyée lors de la tentative d'encodage pour l'interpolation, car aucun exemple représentant une telle mélodie n'a été trouvé dans l'espace latent.

Cela signifie que l'expression du modèle est limitée: toutes les mélodies ne peuvent pas être encodées dans l'espace latent, car la base de données avec laquelle il a été appris ne contenait probablement pas ce type de mélodie.

En sachant cela, il doit être possible d'entraîner des modèles VAE pour manipuler exclusivement des musiques occidentales, ou bien orientales, ou d'autres styles musicaux, en adaptant la base de données d'apprentissage du modèle.

III. Conclusion

MusicVAE est un outil d'assistance de création musical très puissant de part sa capacité à créer des mélodies et des musiques une fois une base de données apprise. Les modèles qu'il propose sont à la fois expressifs, continus, et réalistes, et les interpolations de mélodies qui en résultent sont impressionnantes.

Toutefois, si cet outil est pensé pour les artistes et musiciens, il peut être difficilement dit la même chose pour les développeurs novices, car même en ayant beaucoup cherché sur internet, il ne me semble pas qu'il y ait d'aide de quelque sorte que ce soit pour l'installation ou même la documentation de Google Magenta, ou du moins, l'aide disponible est orienté pour les développeurs très expérimentés.

Je recommande donc à quiconque souhaiterait utiliser Google Magenta de s'aider de personnes à l'aise avec l'installation de modules Python.

Bibliographie

[1] Magenta Tensorflow, https://magenta.tensorflow.org/

[2] MusicVAE: Creating a palette for musical scores with machine learning., https://magenta.tensorflow.org/music-vae

[3] fo40225/tensorflow-windows-wheel: Tensorflow prebuilt binary for Windows, https://github.com/fo40225/tensorflow-windows-wheel

Annexe

Dépôt github (fichier .ipynb et sons) : https://github.com/ocornet/SCIMU

Checkpoints du modèle pré-entraîné disponible au téléchargement ici : https://storage.googleapis.com/magentadata/models/music_vae/checkpoints/cat-mel_2bar_big.tar (1.57 Go) à décompresser dans le même dossier que le fichier .ipynb