Réseaux de neurones appliqués à la musique : Le cas du Jazz

Guilhem Marion

21 mai 2018

Résumé

Ce document propose d'étudier la modélisation et la génération de partitions musicales par apprentissage statistique. A cette fin, nous essaierons de comprendre les enjeux théoriques, musicologiques et mathématiques de cette discipline. Puis, nous nous intéresserons à plusieurs projets existants dont DeepBach, Magenta et WaveNet et nous étudierons leurs spécificités. Nous décrirons alors notre projet DeepJazz qui utilise des réseaux de neurones récurrents afin d'estimer les probabilités d'apparition des notes en fonction de leur contexte grâce à une représentation pertinente des standards de Jazz. De cette distribution nous utiliserons une méthode de Gibbs-Sampling afin de générer des standards de Jazz. L'implémentation [1] a été réalisée en Python et utilise la librairie Keras. Enfin, nous décrirons une méthode utilisant des outils de correction grammaticale permettant d'améliorer les partitions générées.

Mots clefs— Machine Learning, Computer Music, Jazz, RNN, Keras

Contexte Universitaire

Ce projet s'est effectué au laboratoire LIRIS au sein de l'équipe SMA : Système Multi-Agents et dirigé par M. Frédéric Armetta que je remercie chaleureusement pour son implication et son intérêt. Le projet a été proposé par mes soins, né d'intérêts personnels, et construit avec M. Armetta autour de questions d'apprentissage statistiques et d'intelligence artificielle.

1 Introduction

Les premiers travaux de composition musicale assistée par ordinateur remontent aux débuts de l'usage informatique avec comme date emblématique 1956 avec les travaux de Lejaren A. Hiller et Léonard M. Isaacson à l'université de l'Illinois, qui débouchèrent à la suite pour quatuor à cordes *Illiac Suite[Max59]*. Leurs travaux ont consisté à composer un programme à partir d'un système de règles entrées au préalable par un compositeur. Pas vraiment d'intelligence artificielle ici, mais un outil proposé au compositeur ne changeant rien à l'auctorialité de l'oeuvre.

Depuis, l'intelligence artificielle est devenue une discipline en soi, manipulant ses objets d'études de prédilection, mais définissant aussi ses méthodes. L'apparition des réseaux de neurones, avec la puissance de calcul moderne, permet aujourd'hui de dépasser les limites définies jusqu'alors par cette discipline (la chambre chinoise de Jhon Searl[JR80]) et laisse penser pouvoir saisir ce que l'esprit ne pourrait pas saisir par lui même - à l'instar de l'estomac qui ne peut se digérer.

L'idée d'atteindre un modèle cognitif complet n'est alors plus seulement un fantasme de férus de science-fiction. L'idée que l'art puisse être investi par l'intelligence artificielle peut alors être envisagée, mais non pas comme résolution de systèmes de règles définissant un problème combinatoire lié à la musique mais plutôt comme objet capable de création, d'innovation, voire même de sensibilité.

2 Problématiques et fondamentaux de l'apprentissage musical

2.1 Contexte musical

La problématique d'apprentissage musical est une notion liée et aussi vieille que sa formalisation : la musique occidentale s'apprend et s'enseigne et il en faut pour cela un formalisme. Les traités de contrepoints datant de l'époque renaissante et baroque sont encore aujourd'hui épluchés et enseignés afin de comprendre l'évolution du langage et d'en maîtriser les rouages. Il est à noter que la musique européenne a très vite été formalisée : la définition du code musical que nous appelons aujourd'hui solfège a commencé au Moyen-Âge (écriture en neumes), il n'y a alors qu'un pas vers la théorisation et les premiers traités reconnus apparaissent à partir du IX ème siècle.

Dès lors, naissent les premières notions d'apprentis-

1. https ://github.com/GuiMarion/DeepJazz

sage musical. En effet, après des siècles de musicalité de tradition orale, sans formalisme, on cherche à fixer une grammaire déjà existante héritée de nos ancêtres. Il faut alors, à partir d'un corpus donné, chercher des règles, des distributions de notes, des pratiques pour représenter un langage musical. L'idée étant qu'en musique, la théorie suit la pratique.

C'est exactement cette démarche que nous tenterons de suivre dans notre projet : comment à partir d'un corpus de textes musicaux, nous pouvons en extraire de l'information permettant de représenter une grammaire abstraite représentant ces textes.

2.2 Contexte mathématique

Deux approches peuvent se présenter : la formalisation *logique* c'est à dire par un ensemble de règles qui doivent être satisfaites pour que le texte soit valide. Ou bien un ensemble de contraintes probabilistes définissant une mesure conditionnelle dans l'espace des notes. Cette seconde définition semble être la plus riche car permettant d'englober la première, nous la prendrons comme référence.

La question de la conditionnalité (mémoire) semble évidente : l'apparition d'une note peut dépendre des nombreuses notes qui sont apparues avant, c'est le principe des structures mélodiques.

Depuis l'avènement du Machine Learning (et en particulier des réseaux de neurones et leurs applications au Traitement Naturel des Langues), nous avons à notre disposition des outils capables de traiter des objets séquentiels avec une précision d'ordre dynamique. J'entends par là combiner de l'information acquise avec un regard plus ou moins large [2]. Ces modèles sont les Réseaux de Neurones Récurrents, dont la structure interne peut varier, principalement entre Vanilla RNN, dont la structure est la plus simple, puis LSTM et GRU (une variante simplifiée de LSTM[**GRU**]). La capacité des modèles Vanilla RNN à exploiter des dépendances longues a été plus que mise à défaut[T S17], nous utiliserons donc le modèle LSTM pour nos implémentations.

2.3 Génération temporelle

Dans le cas de la génération temporelle la solution la plus évidente est la suivante : on représente des séquences temporelles sous la forme de vecteurs d'entier [3]. On s'intéresse alors à la fonction :

$$f \quad : \quad \mathbb{N}^k \quad \rightarrow \quad \mathbb{R}^n$$

Avec k la taille de la séquence (un LSTM est invariant à la taille des séquences) et n le nombre de symboles de notre grammaire (dans notre cas des notes de musique comprenant la hauteur et la durée).

Cette fonction correspond à un problème de classification (en opposition aux problèmes de régressions) qui à partir du vecteur N nous donne un vecteur D de réels tel que

$$D_i = \tilde{p}(i|N)$$

Et, en d'autres termes, nous donne la *probabilité calculée* d'apparition du i^{me} symbole étant donné le vecteur D.

La phase d'apprentissage correspond alors à estimer f en calculant $\tilde{p}$ pour toutes les séquences et toutes leurs sous-séquences. Cela nous donne la fonction approximée sur notre jeu de données :

$$\tilde{f} \quad : \quad \mathbb{N}^k \quad \rightarrow \quad \mathbb{R}^n$$

L'idée la plus simple est alors de partir d'un symbole de départ que l'on aurait intégré comme premier symbole de toutes nos séquences, générer D^t et construire successivement N^{t+1} avec

$$N^{t+1} = N^t \cdot \underset{i}{\operatorname{argmax}}\ D_i$$

Dans notre réalisation nous utiliserons une méthode plus avancée, utilisant du *sampling* que nous détaillerons plus tard, reposant néanmoins sur le même principe.

3 Réseaux de neurones

Les réseaux de neurones ont été introduits dans les années 50 (perceptron défini en 1958), mais, trop coûteux en calculs, ils n'ont été poussés sur le devant de la scène que très tard. Aujourd'hui, avec l'essor des GPU, ils sont devenus des outils de prédilection du Machine Learning, et permettent de repousser les limites de l'apprentissage statistique.

3.1 Neurone formel

Le neurone formel (cf fig. 1) est à la racine du réseau de neurone, il est défini comme une fonction paramétrée f_ω :

$$f_\omega(x) = a(x \cdot \omega)$$

Où x est l'entrée du neurone, à valeur dans $\mathbb{R}$, ω un vecteur de poids $\{\omega_1, ..., \omega_n\}$ avec $n = |x|$ la taille de l'entrée et a une fonction d'activation, souvent une sigmoïde définie par une fonction logistique ou une tangente hyperbolique.

2. Cette précision semble avoir son importance car les chaînes de Markov peuvent être d'ordre variable, et ainsi récupérer de l'information à différentes échelles, mais ne peuvent combiner l'information des différentes échelles et nécessite d'avoir un ordre fixé, ce n'est pas le cas de LSTM.

3. Le traitement des données est toujours faisable, même si parfois pénible, car il suffit de trouver une bijection entre les objets que l'on manipule (taille fini), et les entiers (infini dénombrable).

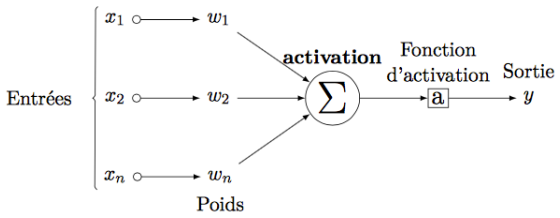

FIGURE 1 – Neurone formel

Une couche de neurones est un ensemble de neurones formels ayant le même nombre d'entrées. On peut donc lui attribuer une matrice de poids contenant le vecteur de poids de tous les neurones la composant. Un réseau de neurones consiste en une succession de couches de neurones dont la première couche correspond à l'injection de l'entrée dans le réseau et la dernière couche la sortie du résultat. La couche n doit avoir autant d'entrées que la couche $n-1$ n'a de sorties. Les couches entre la couche d'entrée et la couche de sortie sont appelées *couches cachées*.

3.2 Perceptrons

Le perceptron[F58] (cf fig.2) est un réseau de neurones contenant une seule couche cachée avec la même fonction d'activation (sigmoïde, non linéaire) pour tous ses neurones, et une couche de sortie linéaire. Il est donc représenté par n la taille de l'entrée, k le nombre de neurones cachés, p son nombre de sorties, W sa matrice de poids et O la fonction d'agrégation de sortie. Il applique donc la fonction :

$$R(x) = Os(W \cdot x) \tag{1}$$

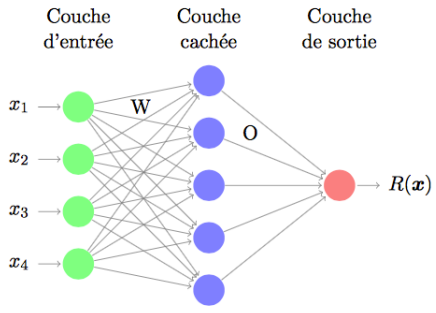

FIGURE 2 – Perceptron

L'idée d'un tel réseau est que l'on peut lui permettre d'approximer une fonction f défini par un jeu d'exemples $E = \{x_i, f(x_i)\}$ en ajustant sa matrice de poids afin de minimiser une fonction de perte [4] L préalablement définie, on utilisera par exemple une distance simple :

$$L(x) = |f(x) - R(x)|^2$$

Le réseau étant entièrement différentiable, on utilise une méthode de descente de gradient afin de calculer cette matrice de poids minimisant la fonction de perte[LeC+12].

3.3 Approximateurs universels

Tout l'intérêt de cette méthode est d'utiliser des perceptrons à plusieurs couches pour approximer f du fait qu'ils aient été démontrés *approximateurs universels*[Hor91] c'est à dire que, pour toute fonction f il existe

$$F(x) = \sum_{i=1}^{N} v_i \varphi\left(w_i^T x + b_i\right)$$

telle que,

$$|F(x) - f(x)| < \varepsilon$$

F étant une fonction définissable par un perceptron multi-couches (cf eq. 1), il est donc possible, quelle que soit la complexité de la fonction à approximer [5], de trouver $\tilde{f}$ tel que

$$|\tilde{f}(x) - f(x)| < \varepsilon$$

Ce qui donne plus que bon espoir dans le fait d'utiliser un réseau de neurones pour la génération musicale.

3.4 Réseaux de neurones récurrents

Les réseaux de neurones récurrents (RNN), permettent de traiter des données séquentielles : $x_1, ..., x_n$ où les x_i sont dépendants dans le temps. Ils maintiennent pour cela une couche cachée faisant évoluer un état interne permettant de simuler un processus de mémoire. Le calcul de $R(x_t)$ se fait donc en fonction de x_t mais aussi de l'état interne qui, lui même, est dépendant des x_i avec $i < t$.

4. Il est à noter que la définition de cette fonction est extrêmement importante car elle va conditionner l'apprentissage, il faut donc pour la définir bien étudier le jeu de données et l'adapter à ce dernier.

5. Cette notion théorique rend, bien entendu, seulement compte de la possibilité d'approximer une fonction avec un perceptron multi-couches, et ne rend pas compte des problématiques d'apprentissage, taille du jeu de données, définition de la fonction de perte, temps de convergence, ...

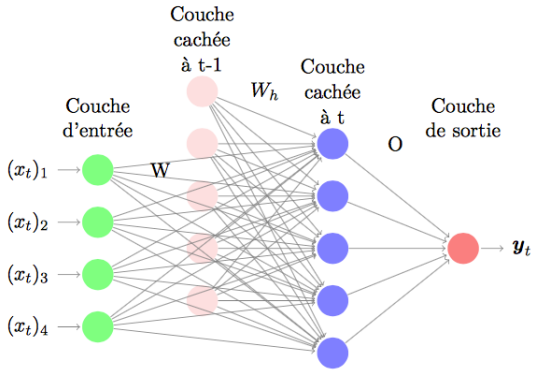

FIGURE 3 – RNN

La figure 3 représente un réseau de neurones récurrents Vanilla RNN, qui est le modèle le plus simple et le plus utilisé aujourd'hui. LSTM, est un peu plus compliqué et nous ne pouvons en décrire le fonctionnement dans ce document.

4 État de l'art

Nous avons désormais posé les bases permettant d'appréhender sereinement l'apprentissage musical, nous allons essayer dans cette partie de faire un état des lieux de la littérature sur le sujet.

4.1 Apprentissage audio

Nous avons, précédemment, évoqué la musique comme un langage représenté par un ensemble de symboles Σ, se réduisant ainsi à la notion d'apprentissage symbolique, très proche des procédés issus de NLP. On peut cependant voir la musique non pas comme une grammaire mais comme du son, incluant ainsi de nombreux genres musicaux alors exclus (musiques électroacoustiques, et toute musique travaillant plus sur le timbre que sur les notes). C'est l'objet d'un projet mené par DeepMind consistant à produire un module *text to speech* (TTC) qui ne soit pas *concatenative TTC* mais *parametric TTC*, en effet une très grande partie des modules TTC utilise une grande base de données contenant des phonèmes (plus petite entité de son) et les combinent afin de former la phrase désirée. Cela rend impossible le changement de voix sans réenregistrer toute la base de données, idem si l'on veut changer les émotions du locuteur, par exemple.

L'idée du projet Wavenet[Oor+16] est de travailler directement sur le signal audio, sample par sample, le modèle est conditionné par la phrase à dire. Le fonctionnement du modèle est exactement le même que décrit dans la partie précédente (on construit N^{t+1} à partir de N^t et de $\underset{i}{\text{argmax}}\ D_i$). Ce projet a aussi eu beaucoup de retombées dans le monde de la génération musicale car il rend possible la génération de fichier audio (résolvant la question de la production de son par l'ordinateur dans le cas de l'apprentissage symbolique). Durant le projet, nous avons effectué des expériences avec ce modèle (que nous ne développerons pas ici, car il est difficile de rendre compte de résultats sonores par écrit) et nous nous sommes rendus compte d'une énorme lacune de la dépendance à moyen et long terme. Il est capable d'apprendre de manière très efficace les timbres des instruments (piano, guitare, synthétiseur, batterie), mais n'est pas capable de construire une phrase musicale. Ce modèle n'est donc pas capable de se représenter une grammaire musicale mais il pourrait être intéressant pour, par exemple, imiter un instrument particulier s'il est conditionné par la partition à jouer.

Il existe d'autres projets utilisant directement des fichiers audio, notamment pour le clustering, cependant ils utilisent des outils issus du traitement du signal pour se représenter les timbres et les échelles harmoniques plutôt que du machine learning[RNR15]. Ils ne permettent donc pas au programme de se représenter une grammaire.

4.2 Apprentissage symbolique

Les techniques d'apprentissage audio n'ont, à ce jour, pas permis de donner de résultats convaincants en apprentissage musical. Néanmoins, de nombreux résultats apparaissent en apprentissage symbolique (manipulation de symboles). Cette approche rend la tâche plus simple du fait que les symboles sont (relativement) aisément manipulables, mais aussi car les travaux en Traitement Automatique des Langues nous donnent de solides exemples.

Il est à noter que l'une des première tâches est la représentation et le traitement des données (que nous détaillerons dans la partie suivante).

4.3 Improvisor

Une méthode très courante -aussi issue du traitement automatique des langues - est l'utilisation de modèles probabilistes. Elle consiste à étudier un large corpus d'un langage, d'en définir l'alphabet et de calculer la probabilité d'apparition de chacune des lettres en fonction des n précédentes : on travaille alors sur des $n-gramme$ ce qui correspond à un modèle de Markov d'ordre n.

Cette démarche en traitement automatique des langues a conduit à la modélisation plus ou moins précise de syntaxes - notamment en langue anglaise, avec de grands corpus comme l'oeuvre de Shakespeare. Si l'on applique ce modèle au langage musical, on modélise la syntaxe commune à tout le corpus, c'est à dire le style musical ou le style de jeu d'un musicien. Cette analyse statistique peut être utilisée pour établir des clusters de compositeurs et ainsi permettre à un programme après l'analyse d'un morceau inconnu d'en déterminer son auteur.

Exemple de modèle de Markov sur une mélodie simple

Nous allons ici construire une chaîne de Markov correspondant à la mélodie suivante, cela permettra de comprendre comment procéder pour de grands corpus.

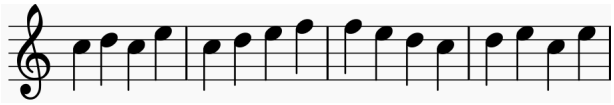

FIGURE 4 – Exemple de mélodie

On a donc quatre états : Do, Ré, Mi et Fa, on calcule maintenant le poids des transitions :

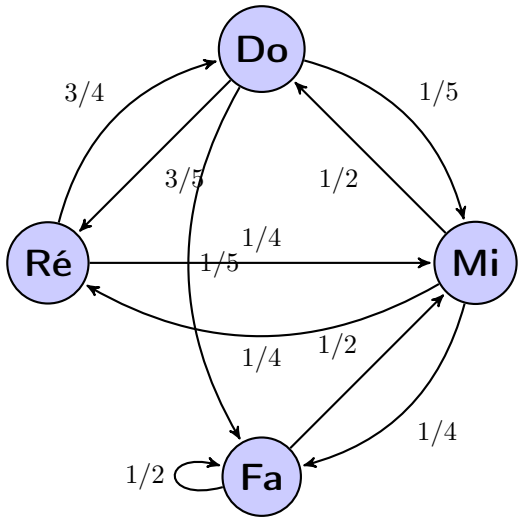

FIGURE 5 – Chaîne de Markov correspondant à la mélodie

A partir de cette chaîne de Markov on déduit une grammaire probabiliste simple qui nous dit par exemple que l'on ne peut pas jouer de Ré après un Fa. Si l'on procède sur de grands corpus les gestes musicaux anecdotiques sont effacés et les gestes idiomatiques quant à eux sont accentués. On modélise donc la syntaxe du corpus. On peut ensuite utiliser ce modèle pour faire de la génération, il suffit de partir d'un état donné et de tirer une variable aléatoire X dans un ensemble E et de définir autant de sous ensemble de E que de transitions. Si $X \in E_i$ alors on choisit la transition i. On génère alors des compositions "à la manière de".

C'est cette méthode que Jon Gillick, Kevin Tang et Robert M. Kellerdes des universités de Standford, Cambridge et Harvey Mudd College ont utilisé[**JazzGram**], en agrégeant dans chaque état les n dernières notes, l'accord courant, la position dans la mesure et la tonalité.

Leur but fut de générer des improvisations de solos de Jazz en utilisant des grammaires probabilistes (premiers travaux par Keller en 2007), l'article présente l'apprentissage automatique de telles grammaires. Ils partent de corpus de plusieurs musiciens de Jazz reconnus (Keith Jarett, Miles Davis, etc ...) et établissent des modèles probabilistes pour chacun d'entre eux afin de capturer les gestes idiomatiques de ces musiciens. En utilisant chacun de ces modèles pour la génération, le programme permet de générer des solos à la manière de célèbres musiciens et ce sur n'importe quelle grille d'accords. Un programme utilisant ce principe fut mis en place par Robert Keller de l'Université de Harvey Mudd College, il s'appelle Impro-Visor et on le trouve aisément sur internet.

Ces travaux peuvent être extrêmement intéressants d'un point de vue musicologique, il permettent d'apprendre rapidement beaucoup de choses sur un style musical ou un jeu de musicien. Les musiciens de Jazz seront par exemple ravis de pouvoir se servir de tels programmes pour récupérer des phrasés intéressants sans avoir à écouter toute la discographie d'un musicien.

4.4 DeepBach

Il existe néanmoins des projets entièrement dédiés à la génération de partitions, c'est le cas des projets utilisant des réseaux de neurones qui permettent, quant à eux, de fournir que de très faibles informations sur les structures internes - une chaîne de Markov s'analyse, pas un vecteur de poids. C'est le cas de l'équipe gérée par François Pachet au CSL (laboratoire d'informatique de Sony à Paris). Il travaille sur les FlowMachines[Pac04], outil privé permettant l'aide à la composition pour les musiciens. Son projet DeepBach[HP16], en licence MIT quant à lui, a fait beaucoup de bruit dans le monde de l'apprentissage musical.

Leur projet est de s'intéresser aux chorals [6] de Bach, ce sont des pièces ponctuées par des points d'orgues (moment où le temps peut s'allonger selon l'interprétation) où l'on retrouve systématiquement des cadences (ponctuation de la phrase musicale, conclusive ou suspensive). Ce point est très important car c'est lui qui permet de ne pas avoir à se représenter une dépense à long terme pour structurer un discours, car la forme est intégrée dans les points d'orgues. De plus, le style utilisé dans les chorals, le contrepoint, est très bien formalisé aujourd'hui, c'est un style aux règles claires et simples.

Les données sont représentées de la façon suivante :

Le temps est quantifié en un seizième de mesure, chaque note est mappée sur un entier. Chaque voix correspond donc à un vecteur d'entier. Une répétition de notes est représentée par le symbole '_', ce qui permet une meilleure aisance d'apprentissage et de génération.

6. Ce sont des pièces polyphoniques baroques à quatre voix, principalement composées par Jean-Sebastian Bach à partir de mélodies sacrées monophoniques et modales.

Un vecteur à deux dimensions est alors utilisé pour nourrir le réseau de neurones qui est construit de la façon indiquée dans la Figure 6.

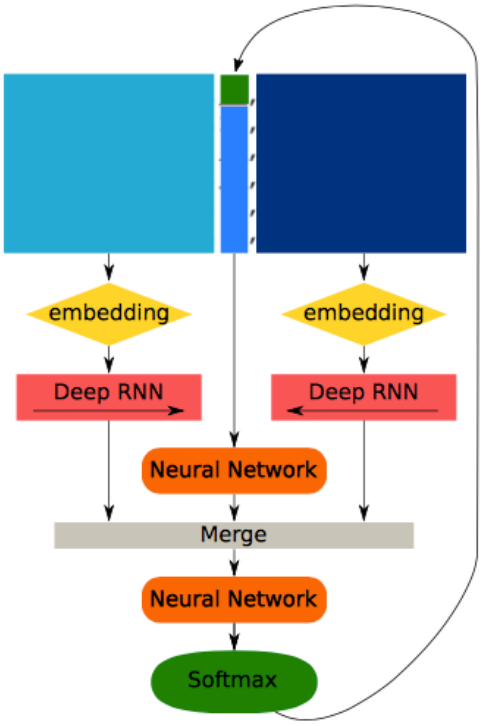

FIGURE 6 – Schéma de l'architecture utilisée.

Le génération est ensuite faite par une méthode de pseudo-Gibbs sampling (expliquée dans la partie projet).

Les résultats sont surprenants, de plus une étude en aveugle, montre que DeepBach est le générateur de chorals le plus convainquant à l'heure actuelle et que 50% des personnes interrogées ne parviennent pas à faire la différence entre une pièce composée par Bach et par DeepBach.

4.5 Magenta

Nous nous sommes intéressés jusqu'ici à la grammaire musicale, mais celle-ci du coté du compositeur, nous avons parlé de notes, d'accords, de théorie, mais jamais de musique, de son. Là est une différence importante : qui fait la musique ? L'interprète ou le compositeur ? La musique savante occidentale fait en effet une grande division entre ces deux mondes : l'interprétation et la composition. Le compositeur invente, il a un fort bagage théorique, il joue avec les notes, non pas sur l'instrument, mais sur le papier, il tente de trouver des agencements, des positions pour faire sonner les instruments entre eux, et, par des notes écrites sur du papier raconter une histoire, trouver une forme musicale. Le travail de l'interprète est tout autre : il doit, en prenant comme appuis intransgressible la partition, la restituer, la donner à entendre de la manière la plus juste possible. C'est un travail artistique à part entière car il doit, comme le compositeur, inventer des techniques de jeu, des positions de doigts, pour faire sonner au mieux cette partition et ainsi raconter une histoire, mais cette fois-ci, plus proche du son que des notes.

Dans une approche du musicale par l'ordinateur, il faut se poser la question de ce que nous voulons modéliser, l'interprétation ou la composition ? Si c'est la musique que nous voulons aborder, ce sont les deux qu'il faudra tenter d'embrasser.

Le format MIDI, propose une façon de représenter la musique non pas comme partition (comme le fait lilypond [7], par exemple) mais comme des gestes, il permet ainsi d'enregistrer la vélocité (force avec laquelle on tape une note), l'agencement temporel (un humain jouera jamais à la milliseconde prés comme indiqué sur la partition, et ce décalage rythmique fait parti du travail d'interprétation). Il existe ainsi de nombreux outils, appelés instruments virtuels, parfois élaborés à l'aide d'outils de Machine Learning [8], permettant de générer du son à partir de séquences midi ; ces outils peuvent être utilisés pour présenter un projet entièrement informatique. Étant donné que le format midi peut contenir des notes et des gestes, nous pouvons l'utiliser afin d'apprendre à jouer de la musique, et de condenser ces deux aspects : c'est le projet de Magenta Performance RNN[SO17].

Leur projet utilise un réseau de neurones récurrents composé de cellules LSTM et ils utilisent la base de données *Yamaha e-Piano Competition dataset* [9] proposant plus de 1400 performances de pianistes professionnels sur pianos électriques enregistrées au format midi. Cette base de données permet donc de se représenter la grammaire musicale de la musique savante occidentale composées pour piano, rendant le jeu de données homogène selon ce critère, mais aussi de se représenter ce qu'est une bonne interprétation de cette musique. Leur représentation des données est la suivante :

- 128 *note-on events*, pour chaque note interpretable par la norme midi. Cet événement correspond au fait d'appuyer sur une touche du piano, c'est un événement natif du format midi.
- 128 *note-off events*, la même chose que *note-on* mais pour le fait de relâcher la touche, c'est aussi un événement natif du format midi.
- 32 *velocity events*, correspondant à la force avec laquelle est frappée la touche, décomposée en 32 possibilités.

Ces événements sont quantifiés à 10 ms près, et sont représentés dans un *vecteur one-hot* ce qui donne une large palette pour l'interprétation temporelle, de plus quand il

7. Éditeur de musique en code, proche de latex.
8. WAVES plug-ins ou certaines utilisations du projet WaveNet présenté plus tôt.
9. http ://www.piano-e-competition.com

y a n'y a pas d'événement on garde les notes déjà jouées, ce qui permet une plus grande malléabilité lors de l'apprentissage.

La phase de pré-traitement consiste à transposer tout le jeu de données, pour avoir une représentation de toutes les tonalités, ainsi qu'un *time-stretching* afin de se représenter les différents tempos. Un paramètre supplémentaire : *temperature* permet , lors de la génération, de jouer sur le PRNG [10] afin de contrôler un petit peu la génération.

Ce projet donne de très bons résultats dans ce contexte bien particulier du répertoire pianistique de musique savante occidentale, de nombreux exemples peuvent être trouvés sur la page internet de Magenta. Cependant, comme de très nombreux, si ce n'est l'intégralité, des projets d'apprentissage musical, le programme n'arrive pas à se représenter les dépendances long termes et ainsi à construire une forme. Des exemples d'une durée de 5 minutes sont présentés pour s'en rendre compte, la cohésion musicale arrive, en général, à tenir 30 secondes.

5 Projet

Afin de mieux se représenter le problème et ses détails d'implémentation nous avons décidé de réaliser un module d'apprentissage musical. Nous avons choisi le cas particulier des standards de Jazz.

5.1 Définition du problème

Nous avons vu plus tôt l'apprentissage de grammaires anciennes, liées à la notion de contrepoint, notion très bien formalisée à l'époque baroque. Notre projet sera tout autre, représenter une grammaire mal définie par les musicologues : le Jazz. Le Jazz fait effectivement débat dans le monde de la musicologie, étant que partiellement formalisable par les outils issus de la musique classique. Sa définition, éminemment dynamique, liée au fait de sa nature improvisatrice en rend une formalisation figée très difficile. Cependant, la reconnaissance à l'oreille d'un style Jazz n'est pas si difficile par un humain, on a donc la grande intuition qu'il existe une grammaire sous-jacente, seulement trop complexe pour être formalisée par les outils musicaux actuels. De cette idée naît le projet d'utiliser des réseaux de neurones récurrents et leur propriété d'approximateurs universels afin de tenter de représenter (à l'échelle du RNN) la grammaire du Jazz.

Un standard de Jazz (cf fig.2) est défini par une mélodie (appelé thème), souvent jouée aux soufflants (instruments à vent) et d'une grille d'accords permettant de l'accompagner. Le principe d'un morceau de Jazz est le suivant : le soliste joue le thème en étant accompagné avec les accords du standards, une fois le thème joué chaque soliste (soufflants, piano, guitare) improvise un solo accompagné par les autres musiciens sur les mêmes accords du standards. Une composition de Jazz est donc définie par cette mélodie accompagnée.

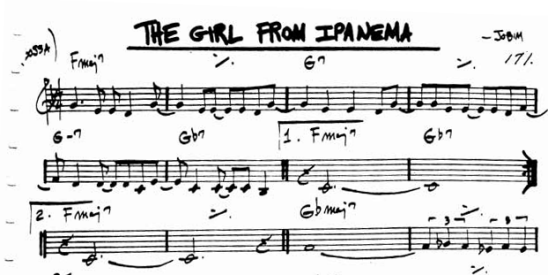

FIGURE 7 – Un extrait de The Girl From Ipanema par Tom Jobim, extrait du RealBook

5.2 Représentation et traitement des données

Le première tâche est de définir la représentation des données. Il faut, pour que le réseau de neurones puisse travailler, lui fournir un vecteur de dimension k, il est préférable que ces vecteurs soient à valeur dans $\mathbb{Z}$. Nous avons donc choisi de représenter les standards par des vecteurs à 2 dimensions, une dimension mélodique et une dimension harmonique (les accords).

La transformation pour la mélodie est la suivante : nous calculons l'ambitus [11] maximal dans notre jeu de données. Nous définissons alors l'intervalle dans lequel nous allons travailler et nous affectons chaque note à un entier séquentiellement et conjointement : si DO2 est la note la plus grave du jeu de données elle sera affectée à 1, 10 sera donc LA2 et 11 à LA#2. Cela permet au réseau de neurones de se représenter le concept de hauteur et de distance entre les notes, il peut ainsi les comparer deux à deux. La limite de cette représentation est le fait qu'il ne puisse pas se représenter l'arithmétique modulaire qu'il y a en musique (DO1 et DO2 représentent la même note à une octave de différence). Nous espérons que le réseau de neurones puisse néanmoins apprendre que $x + 12$ est très proche de x du fait qu'il est à distance d'octave, nous avons bon espoir car cette notion est à la base de l'harmonie tonale utilisée dans le jeu de données.

Les accords, quant à eux, utilisent le même principe. On effectue le produit cartésien de l'ensemble des notes (de cardinal 12) et de l'ensemble de ce que l'on appellera *couleur* c'est à dire la nature de l'accord, par exemple majeur, mineur. En jazz, la notion de nature d'accord, s'étend à la notion de couleur dans le sens où l'enchaînement des accords ne fait pas que structurer le discours musical, comme en musique classique, mais est aussi là pour le colorer. Il existe donc un très grand nombre de

10. Générateur de nombres pseudo-aléatoires.

11. distance entre la note la plus aigu et la note la plus grave

couleurs, dans notre jeu de données il en existe 46, les accords sont donc à valeur dans [1, 552]. Les accords sont distribués dans [1, 552] selon leur note fondamentale, puis selon leur couleur [12], cela permet au réseau de neurones de se représenter la notion de fondamentale/couleur.

Pour la représentation du rythme, nous avons choisi de quantifier le temps à la double croche (quart de temps), les durées de notes utilisent un symbole spécial, l'underscore _, il permet de signifier qu'une note est répétée, cela permet de rendre beaucoup plus évidente la génération.

A cela sont ajoutés les symboles de début et fin de séquences.

L'extraction de ces informations, paraissant d'assez bas niveau, est, en fait, relativement complexe. Il faut en effet traiter les fichiers midi qui contiennent des notes (souvent en polyphonie) comme des événements TouchON et TouchOFF qui correspondent au fait d'appuyer sur la touche d'un piano et de la relâcher. Il faut donc faire un traitement de données pour extraire les informations de durée, mais surtout d'accords. Il faut en effet être capable, à partir de vecteurs de notes, de renvoyer une couleur et une fondamentale. Pour cela nous avons utilisé la librairie *music21* très incomplète, mais surtout codé notre propre module d'identification d'accord [13] pour notre jeu de données. Notre jeu de données est issu du projet *openbook* [14] qui recense des partitions de Jazz au format lilypound pour en faire un éditeur de pdf modulable.

5.3 Modèles d'apprentissage

Nous avons réutilisé le modèle d'apprentissage utilisé pour DeepBach (cf fig.1), le principe est d'utiliser un réseau de neurones pour calculer la distribution de probabilités d'apparition de chaque note en fonction de son environnement musical, la génération n'est donc pas faite par le réseau mais par sampling. Le principe est donc de capter de l'information structurelle du passé et du futur mais aussi de l'environnement harmonique. On utilise donc deux réseaux récurrents, un pour le passé, un pour le futur, et un réseau non récurrent pour l'harmonie (cette note va-t-elle bien avec cet accord ?). Le réseau apprend donc la probabilité d'apparition de chaque note selon son contexte. Le procédé est le même pour les accords.

On utilise ensuite le réseau en mode génératif pour construire la fonction $p1(n, C)$ qui donne la probabilité d'apparition de la note n en fonction du contexte C et $p2(c, C)$ pour la probabilité d'apparition de l'accord c en fonction du contexte C.

12. nous avons tenté de rapprocher les couleurs sémantiquement proches Majeur et Majeur7 par exemple

13. https ://github.com/GuiMarion/MusicChordExtraction

14. https ://github.com/veltzer/openbook

5.4 Génération

La génération se fait par Gibbs Sampling. Le principe est de remplir aléatoirement un vecteur et ensuite de tirer aléatoirement une note et de la re-échantillonner selon p calculé par le réseau de neurones, cette étape est à effectuer N fois, N dépendant de la précision désirée. Cela permet d'avoir une distribution finale très proche de la distribution attendue, en prenant en compte le passé, le futur, et l'harmonie. Cela se rapproche des techniques de composition *in vivo* où le compositeur compose en prenant en compte tous les paramètres en même temps, et non pas de façon séquentielle voix par voix. Cette méthode est intuitivement bien plus efficace que la méthode décrite dans la première partie qui consiste à construire séquentiellement en prenant la note qui maximise p par rapport au passé.

Algorithm 1 Sample une séquence temporelle

Require: S une séquence temporelle, $P()$ la distribution des notes, N la durée du sampling.

1: S est initialisé aléatoirement
2: **while** $N \neq 0$ **do**
    On choisit aléatoirement un temps t et on resample la note au temps t avec $P(n, t|S)$
    $N \leftarrow N - 1$
3: 4: **end while**

Construisons alors un vecteur d'entiers de deux dimensions correspondant à un morceau, il suffit alors de le retransformer grâce à la fonction de Mapping définie dans *Représentation et traitement des données*, si l'on veut pouvoir écouter le résultat, il faut transformer le vecteur en fichier midi, pour cela nous avons élargi les fonctionnalités de notre projet *MusicChordExtraction*.

5.5 Résultats

Le projet présente de très bons résultats car il permet de générer des standards de Jazz *dans le style* (cf fig.3, todo mettre une image). Il possède une grande compréhension de la structure du discours musical, en ayant une bonne utilisation des cadences et des accords de dominante. Les successions d'accords sont naturelles et possèdent une couleur Jazz, cependant il reste des lacunes dans la vision de la forme, problème que l'on retrouve très souvent dans la littérature et qui est évité par DeepBach en fixant les début et fin de phrases (très courtes) en imposant les points d'orgues.

Les mélodies quant à elles, sont de qualité plus aléatoire, mais manquent souvent elles aussi de structure à long terme.

Les formes classiques du Jazz (AABA, AABB, ...) ne sont globalement pas comprises par le programme. Une solution serait de fixer la forme des morceaux ou d'ajouter des labels de forme dans le jeu de données.

Étant donné la courte durée du projet, nous n'avons pas pu mettre en place de procédé permettant d'obtenir des résultats quantitatifs. Mais nous pensons continuer à travailler sur le projet et mettre en place, entre autre, un test psychologique permettant de déterminer si une personne est capable de reconnaître un morceau composé par le programme d'un morceau original.

5.6 Apprentissage des standards de Jazz

Ce projet se place, selon nous, dans la continuité des projets en apprentissage musical. Tous les projets déjà menés en apprentissage musical sur le Jazz[**JazzGram**] [15] se basant sur des fichiers midi de jeu musical et souvent pianistique, notre projet semble donc tout à fait innovant. Il est, à notre connaissance, le premier à travailler sur les standards de Jazz comme objet d'étude. Et semble être, de plus, l'un des plus avancé d'un point de vu théorique, utilisant les outils les plus récents en terme d'apprentissage et de génération.

5.7 Correction de partitions

Étant donné que nous avons comme objectif de continuer à travailler sur ce projet en dehors du projet *POM*, nous aimerions ajouter une partie utilisant les techniques de corrections grammaticales[CT16] NLP pour affiner la génération. On retrouve en effet, très souvent dans la littérature des procédés de corrections grammaticales *invariant à la langue*[Hua+18], c'est à dire qui ne dépendent que de la langue du jeu de données, et peuvent donc s'appliquer très simplement à la musique.

On verrait cela comme l'apprentissage d'une fonction :

$$f \quad : \quad \mathbb{N}^n \quad \rightarrow \quad \mathbb{N}^n$$

qui appliquerait une correction à une partition P afin de la rendre grammaticalement correcte selon certains critères, cette correction peut être nulle. Une telle fonction peut s'apprendre de différentes façons, soit à partir d'un jeu de données grammaticalement correct, dans ce cas il n'y a que peu de chances d'améliorer P car elle a été générée par sampling à partir de ce même jeu de données. La fonction peut aussi être apprise à partir de jeux de corrections. L'idée serait alors la suivante : on sollicite l'aide d'un musicien averti (enseignant au conservatoire par exemple) afin de corriger un certain nombre de partitions générées par le programme et on apprend f à partir de ce jeu de correction. On peut alors avoir espoir de corriger des partitions générées selon le critère de validité de cet enseignant (à priori d'excellente qualité). De plus étant donné que le jeu de corrections à été fait à partir des partitions générées par le programme, on a bon espoir de calculer f avec un jeu de données raisonnablement grand. L'idée intéressante est que ce processus peut être récursif : on génère beaucoup de partitions, on les corrige et on peut les ajouter au jeu de données du premier réseau qui va alors générer de nouvelles partitions (à priori meilleures), on peut alors apprendre f' à partir de ce jeu de données, et recommencer le processus jusqu'à ce que f devienne stable.

15. https ://deepjazz.io

6 Conclusion

Ce document a permis de présenter le contexte scientifique de la génération de musique par ordinateur. Nous avons fait le point sur les outils mathématiques à manipuler et avons montré que les réseaux de neurones récurrents présentent des qualités qui en font de très bon candidats. Nous avons ensuite fait le tour de littérature sur le sujet et avons montré que les dépendances moyen et long termes constituent la problématique majeure de la discipline et qu'il n'existait, à ce jour, aucun travaux sur la génération de standards de Jazz. Nous avons donc implémenté un module qui en s'inspirant de la littérature faisant état de l'art nous a permis de générer des standards de Jazz à partir d'une base de données de fichiers midi. Ce travail permet de montrer qu'il est possible d'utiliser des réseaux de neurones profonds pour modéliser des grammaires complexes avec une disparité interne relativement grande. Nous n'avons cependant pas pu dépasser le problème de la dépendance temporelle, mais avons des idées permettant, dans les améliorations futures, de tenter de le dépasser. Finalement, nous avons introduit une méthode permettant d'améliorer la qualité de la génération de réseaux de neurones profonds en appliquant des outils de correction grammaticale, demandant néanmoins l'intervention d'un humain qualifié pour son apprentissage.

Bibliographie

[CT16] Liu Chien-Hung et Chuan-Kang Ting. "Computational Intelligence in Music Composition : A Survey". In : PP (déc. 2016), p. 1–1.

[F58] Rosenblatt F. "The perceptron : a probabilistic model for information storage and organization in the brain". In : *Psychological Review* (1958).

[Hor91] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In : *Neural Networks* 4.2 (1991), p. 251–257. issn : 0893-6080. doi : https://doi.org/10.1016/0893-6080(91)90009-T. url : http://www.sciencedirect.com/science/article/pii/089360809190009T.

[HP16] Gaëtan Hadjeres et François Pachet. "DeepBach : a Steerable Model for Bach chorales generation". In : *CoRR* abs/1612.01010 (2016). arXiv : 1612.01010. url : http://arxiv.org/abs/1612.01010.

[Hua+18] An-Ta Huang et al. "Discovering Correction Rules for Auto Editing". In : (mai 2018).

[JR80] Searle J.R. "Minds, brains and programs." In : *The Behavioral and Brain Sciences* (1980).

[LeC+12] Yann A. LeCun et al. "Efficient BackProp". In : *Neural Networks : Tricks of the Trade : Second Edition*. Sous la dir. de Grégoire Montavon, Geneviève B. Orr et Klaus-Robert Müller. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, p. 9–48. isbn : 978-3-642-35289-8. doi : 10.1007/978-3-642-35289-8_3. url : https://doi.org/10.1007/978-3-642-35289-8_3.

[Max59] Hille Lejarenn Isaacson Leonard M. (Leonard Maxwell). *Experimental music ; composition with an electronic computer*. The Behavioral et Brain Sciences, 1959.

[Oor+16] Aäron van den Oord et al. "WaveNet : A Generative Model for Raw Audio". In : *Arxiv*. 2016. url : https://arxiv.org/abs/1609.03499.

[Pac04] F. Pachet. "On the Design of Flow Machines". In : *A Learning Zone of One's Own*. Sous la dir. de M. Tokoro et L Steels. The Future of Learning. Amsterdam : IOS Press, 2004.

[RNR15] Maxime Robin, Grégoire Nicolle et Alexandre Rota. "Automatic sounds clustering approach based on a likelihood measure computation". In : (mai 2015).

[SO17] Ian Simon et Sageev Oore. *Performance RNN : Generating Music with Expressive Timing and Dynamics*. https://magenta.tensorflow.org/performance-rnn. Blog. 2017.

[T S17] V. Gripon. T. Stérin N. Farrugia. "RNNs : An Intrinsic Difference Between Vanilla And GRU Models." In : *COGNITIVE@IARIA, Athens, Greece,* (2017).