DeepBach: a Steerable Model for Bach Chorales Generation

Eléonore Bartenlian, Adrien Pavao

1 Introduction

Cet article présente DeepBach[1], un outil orientable permettant de générer de la musique polyphonique semblable aux chorals de Bach.

Le corpus est constitué de pièces d'environ une minute chacune pour 4 voix (soprano, alto, tenor, basse). Elles sont écrites selon un principe commun : le compositeur prend une mélodie connue et l'harmonise avec les trois voix graves qui accompagnent la soprano. Un des défis principaux était que la composition devait être cohérente à la fois entre les harmonies (les notes chantées simultanément) et au sein d'une même voix (l'évolution des voix au cours du temps).

Une première approche a été proposée par Ebcioglu [2] en 1988. Le problème était vu comme un problème de satisfaction de contraintes dans lequel 300 règles énoncées à la main caractérisaient le style de Bach. Cependant, cette méthode est coûteuse et ne produit pas de bons résultats. Les premières solution avec réseaux neuronaux sont apparues plus tard (Hild et al. [3] ; Allan et al.[4]) mais les résultats n'étaient pas convaincant non plus. Récemment (Boulanger-Lewandowski et al.[5] ; Mikolov et al.[6] ; Chung et al.[7] ; Lyu et al.[8]), des approches agnostiques (ne nécessitant pas de connaissances du style de Bach) avec réseaux neuronaux ont été explorées, mais celles-ci manquaient de flexibilité et étaient trop générales pour réellement saisir le style de Bach. L'avancée la plus récente est BachBot (Liang Feynman[9]), un réseau LSTM spécialisé dans les musiques de Bach qui génère une harmonisation de bonne qualité mais qui n'est néanmoins pas flexible.

2 DeepBach

2.1 Représentation des données

Les auteurs de l'article ont utilisé un encodage MIDI pour représenter les notes (une note est symbolisée par un chiffre). Un temps est divisé en 4 parts égales ; une voix est donc représentée par une liste de notes. Les rythmes ont été modélisés en ajoutant un symbole de retenue "_" qui code le fait qu'une note soit maintenue.

La partition d'une musique comporte plus d'informations que les notes chantées (les paroles, la clef, la mesure, le chiffrage, les points d'orgues, etc.). Les auteurs ont pris le parti de ne garder que les points d'orgues et les sous-divisions des temps. Pour ce faire, ils ont ajouté les métadonnées suivantes :

- La liste des points d'orgues F qui indique s'il y a un point d'orgue ou non au dessus d'une note. Dans les chorals de Bach, ils indiquent une fin de phrase et sont spécifiques à son style de composition.
- La liste des subdivisions S qui contient les indices de subdivision des temps (des entiers entre 1 et 4), et qui joue le rôle de métronome.

Ainsi, un choral est représenté par le couple (V, M) composé des voix et des métadonnées. Pour les chorals de Bach, V est donc une liste de 4 voix et M une collection des deux listes F et S.

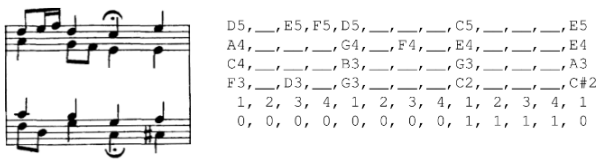

Figure 1 – Extrait d'un choral de Bach et sa représentation.

Par ailleurs, au lieu de transposer tous les chorals dans la même tonalité comme dans les méthodes classiques, les auteurs les ont transposés de toutes les manières possibles et sont ainsi passés d'un corpus de taille 352 à 2503. Cela permet notamment d'avoir une plus grande diversité lors de la génération : si tout était transposé dans une même tonalité, certaines configurations d'accords ne pourraient pas apparaître.

2.2 Architecture du modèle

Les auteurs définissent un réseau de dépendance sur les variables de V à l'aide d'un ensemble de probabilités conditionnelles paramétrisé par $\theta_{i,t}$: $\{p_{i,t}(V_i^t|V_{\setminus i,t}, M, \theta_{i,t}\}_{i\in[4],t\in[T]}$, où les métadonnées M sont données, T est la taille du choral considéré, V_i^t est la voix i au temps t et $V_{\setminus i,t}$ toutes les variables de V^t sauf V_i^t. Le modèle est invariant dans le temps c'est-à-dire que $\theta_i := \theta_{i,t}$, $p_i := p_{i,t}$ $\forall t \in [T]$. Enfin, la log-likelihood est maximisée, ce qui équivaut à résoudre 4 problèmes de classification dont le but est de prédire une note en connaissant la valeur des notes voisines, la subdivision des temps courante et la présence ou l'absence de points d'orgue.

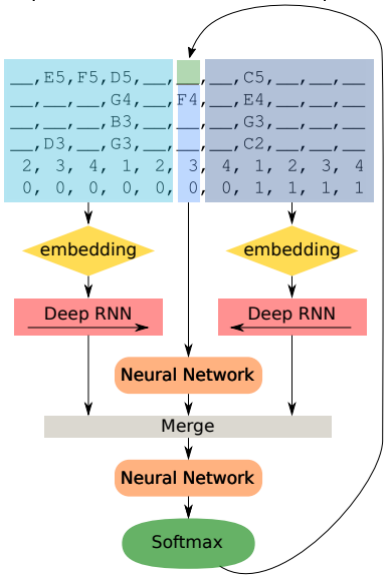

Figure 2 – Représentation graphique du réseau neuronal pour la prédiction soprano p1.

Chaque classifieur est modélisé à l'aide 4 réseaux neuronaux : 2 Deep RNN (un qui résume l'information passée et l'autre l'information future), avec en plus un réseau non récurrent pour les notes simultanées. Les sorties de ces trois réseaux son ensuite réunies et passées à un quatrième réseau neuronal (voir Figure 2) qui renvoie $p_i(V_i^t|V_{\setminus i,t}, M, \theta)$. Ces choix d'architectures s'apparentent à la façon dont Bach composait : en effet, quand

on réharmonise une mélodie, il est souvent plus simple d'écrire la musique "à l'envers".

Les interactions entre les notes sont uniquement locales (dépendance à court terme), c'est à dire que seuls les éléments dont l'indice temporel t est entre $t-\Delta t$ et $t+\Delta t$ sont pris en compte par le modèle. Les résultats obtenus l'ont été avec $\Delta t = 16$.

2.3 Génération

La génération des chorals a été faite en utilisant un échantillonnage pseudo-Gibbs. C'est un algorithme de Méthode de Monte-Carlo par chaînes de Markov (MCMC). Durant un nombre d'itérations fixé, on sample la note de la voix i à l'instant t selon la loi de probabilité $p_i(V_i^t|V_{\setminus i}^t, M, \theta_i)$ estimée par le modèle. Les voix et les temps sont parcourus de façon uniforme.

L'utilisation du pseudo-Gibbs plutôt que le Gibbs classique permet d'éviter les problèmes liés à la variabilité des données. En effet, les données utilisées pour l'inférence ne sont pas fixes puisqu'elles sont samplées en cours de route.

Cette méthode présente un avantage intéressant : l'utilisateur peut imposer n'importe quelle métadonnée (par exemple choisir les points d'orgue pour imposer les fins de phrases musicales, imposer certains accords, fixer un rythme ou encore restreindre les tessitures des voix).

3 Résultats expérimentaux

La qualité du modèle a été évaluée sur des volontaires avec des expériences qui s'apparentent au test de Turing.

Les résultats de DeepBach ont été comparés avec avec deux autres modèles : un Maximum Entropy Model[10], et un Perceptron Multicouches entraînés avec un même Δt.

Les participants devaient donner des informations sur leur expertise musicale et choisissaient leur catégorie entre : (1) J'écoute peu de musique, (2) Amateur de musique ou musicien, (3) Étudiant en musique ou musicien professionnel. Les extraits montrés étaient des mélodies hamonisées par les trois modèles et par Bach lui-même. Les participants devaient dirent s'ils pensaient que la musique avait été composée par Bach ou par un ordinateur.

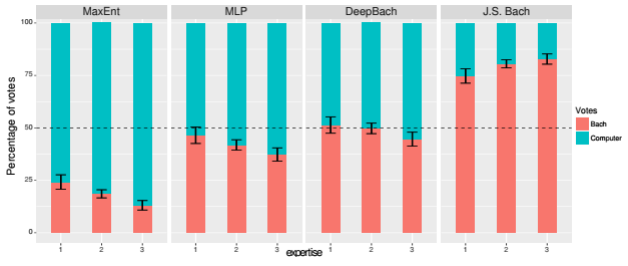

Figure 3 – Résultats de l'expérience : distribution des votes entre "Ordinateur" (en bleu) et "Bach" (en rouge) pour chaque modèle et chaque niveau d'expertise des participants.

On observe que pourcentage de réponses "Bach" augmente lorsque la complexité du modèle augmente. Plus la personne est experte, mieux elle détecte la différence entre les faux et les vrais Bach. Finalement, 50% des participants pensent que la musique composée par DeepBach provient de Bach.

Les auteurs ont également observé que plus de la moitié des chorals composés par DeepBach avaient une majorité de vote considérant qu'ils avaient été composés par Bach, alors que cette valeur n'était que d'un tiers pour le modèle MLP.

4 Composition interactive

Les auteurs ont développé un plugin pour le logiciel MuseScore [1] permettant d'utiliser DeepBach. On peut ainsi générer un choral à partir de rien, ou encore réharmoniser une mélodie.

Pour cela, ils ont effectué quelques changements de représentation des notes : au lieu d'utiliser un encoding MIDI, ils utilisent les noms des notes (par exemple $C\#3$, $Db3$ ou $E\#4$). C'est utile pour les *notes enharmoniques*, c'est-à-dire les notes qui sonnent pareil mais qui sont écrites différemment comme $F\#$ et Gb. Ces notes apparaissent dans un contexte différent du point de vue du machine learning ; cela permet au modèle de générer des notes avec la bonne "orthographe", qui est importante lorsque l'on regarde la partition et pas seulement le rendu audio.

Ils ont également rajouté la signature de la clé courante (c'est à dire les ♯ ou les ♭, une liste K) aux métadonnées. Cela permet à l'utilisateur d'imposer des modulations ou des changements de clé.

5 Discussion

Dans l'ensemble, le modèle semble plutôt bon et meilleur que les tentatives précédents pour imiter le style de Bach. Le point fort principal de la méthode est la possibilité pour les utilisateurs d'imposer des contraintes. Cela permet également de rendre accessible la composition de musique polyphonique à des non-spécialistes, et ouvre une certaine forme de dialogue entre l'ordinateur et l'humain. De plus, le modèle est applicable à n'importe quelle musique polyphonique : il suffit d'entraîner le réseau sur d'autres données. A l'avenir, les auteurs prévoient d'améliorer l'interface, d'accélérer la génération et de faire en sorte que le modèle puisse apprendre sur des petits corpus.

6 Critiques et conclusion

Le problème principal de cette approche (qui est récurrent dans la plupart des modèles de génération de données) et le fait que l'on doive s'en remettre au jugement humain pour comparer les chorals de Bach et ceux de DeepBach. En effet, il n'existe pas de métrique actuellement pour ce problème ; on n'a aucune formule mathématique permettant de quantifier la ressemblance à Bach d'un choral de DeepBach. La mesure de perplexité pour tester la qualité des harmonies générées par rapport aux harmonies de Bach pourrait être une direction intéressante.

Par ailleurs, si l'on veut créer un modèle qui soit le plus parfait possible et quasi indiscernable de la musique de Bach, la représentation actuelle des chorals ne suffit pas. En effet, certains aspects importants ne sont actuellement pas pris en compte ; on peut notamment citer les silences ou encore les nuances qui sont des éléments essentiels participant au caractère authentique d'une musique. Même si, pour les chorals de Bach, les métadonnées actuelles suffisent à obtenir de bons résultats, aucun test n'a été fait pour déterminer si le modèle est convaincant lorsqu'il est entraîné sur d'autres corpus. Dans l'idéal, il faudrait étendre ces métadonnées si l'on voulait pouvoir imiter d'autres compositeurs ou générer d'autres styles et pour rendre la composition assistée plus flexible.

1. `https://musescore.org/fr`

Références

[1] Gaetan Hadjeres et al. "DeepBach : a Steerable Model for Bach Chorales Generation". In : *arXiv :1612.01010v2* (2017). URL : https://arxiv.org/pdf/1612.01010.pdf.

[2] Ebcioglu Kemal. "An expert system for harmonizing fourpart chorales." In : *Computer Music Journal* (1988). URL : http://www.jstor.org/stable/3680335.

[3] Hild et al. "Harmonet : A neural net for harmonizing chorales in the style of js bach." In : *Advances in neural information pro- cessing systems* (1992).

[4] Allan et al. "Harmonising chorales by probabilistic inference." In : *Advances in neural information processing systems* (2005).

[5] Boulanger-Lewandowski et al. "modeling temporal dependencies in high-dimensional sequences : Application to polyphonic music generation and transcription.." In : *ICML-2012* (2012).

[6] Mikolov et al. "Learning longer memory in recurrent neural networks." In : *arXiv preprint arXiv :1412.7753* (2014).

[7] Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." In : *arXiv preprint arXiv :412.3555* (2014).

[8] Lyu et al. "Modelling high-dimensional sequences with lstm-rtrbm : application to polyphonic music generation". In : *Proceedings of the 24th International Conference on Artificial Intelligence* (2015).

[9] Liang Feynman. *Bachbot*. 2016.

[10] Hadjeres Gaetan et al. "Style imitation and chord invention in polyphonic music with exponential families." In : *arXiv preprint* (2016). URL : https://arxiv.org/pdf/1609.05152.pdf.