ANR-14-CE24-0002-01

Projet DYCI2,
WP2 Apprentissage interactif de structures musicales,
WP2.2 Sélection de dimensions et apprentissage par renforcement

Rapport de livrables :

L2.2.1 Sélection de dimensions et apprentissage par renforcement, première version algorithme
L2.2.2 Sélection de dimensions et apprentissage par renforcement, version finale algorithme

Livrable	Date	Contributeurs	Rédacteurs	Contenu
L2.2 Version 01	Février 2018	R. Decelle, K. Déguernel, N. Libermann, E. Vincent (Inria), G. Assayag (Ircam-STMS)	E. Vincent	Maquette Logicielle, Rapport scientifique

Résumé

Ce document regroupe les livrables L2.2.1 et L2.2.2. Il décrit les contributions de la thèse de Nathan Libermann et du stage de Rémi Decelle concernant a) la modélisation de séquences musicales par réseaux de neurones récurrents et b) l'utilisation de réseaux de neurones pour détecter des notes erronées dans l'improvisation en cours et la modification de l'apprentissage pour en tenir compte dans une boucle perception / action.

Adresse du livrable logiciel

DYCI2_WP2_L2.2.zip
sur
https://forge.ircam.fr/p/Dyci2/

De la sélection de dimensions à la détection de fausses notes

Le message musical émis par un improvisateur est porté par une ou plusieurs dimensions « sémantiques ». Par exemple, dans les styles traditionnels, la hauteur peut être porteuse d'information (ni trop déterministe ni trop aléatoire) alors que les timbres sont non informatifs (très déterministes ou très aléatoires). Dans une improvisation contemporaine ou dans certains styles des musiques du monde le timbre peut au contraire devenir prépondérant. Dans notre programme initial de recherche établi en 2014, nous avions prévu d'identifier automatiquement les dimensions sémantiques utilisées par les autres musiciens afin d'adapter la génération. L'apparition du *deep learning* et son utilisation pour l'improvisation musicale, rendue populaire par le projet Magenta[1] de Google rendu public en juin 2016, ont bouleversé la façon de poser le problème. En effet, comparé aux autres méthodes d'apprentissage automatique qui nécessitent d'extraire et de modéliser séparément les différentes dimensions, le *deep learning* a la capacité à traiter conjointement toutes les dimensions. Plutôt que d'identifier les dimensions sémantiques, le problème devient de détecter les « fausses notes » dans une séquence multidimensionnelle, sans viser à identifier explicitement la ou les dimensions erronées, afin d'adapter la génération dans une boucle perception/action. Ce travail a été conduit dans le cadre du stage de fin d'études d'ingénieur de Rémi Decelle [1] et des 18 premiers mois de thèse de Nathan Libermann.

Détection de fausses notes

La première étape a consisté à définir la notion de fausse note. Dans un cadre d'apprentissage automatique, il s'agit d'une note qui s'écarte de la distribution statistique des données.

Dans un premier temps, nous avons considéré la mélodie jouée par le système d'improvisation automatique lui-même, sans prendre en compte les autres musiciens. Afin de pouvoir détecter les fausses notes, nous avons considéré deux types de données d'apprentissage. À partir d'un ensemble de mélodies originales constitué d'improvisations de Charlie Parker (*Omnibook*) ou de mélodies extraites du corpus de musique classique utilisé dans les expériences précédentes (cf. Livrables 2.1.1, 2.1.2) et segmentées en séquences de 9 notes, nous avons :

- sélectionné la moitié des séquences sans modification
- modifié une note de façon aléatoire dans les autres séquences.

Le problème devient alors de prédire si une séquence test (non observée à l'apprentissage) suit la distribution des données originales ou si elle a été modifiée. Cette façon de poser le problème se rapproche du *deep learning* adversarial, qui constitue l'état de l'art actuel pour la génération d'images et qui repose sur un réseau de neurones appelé discriminateur visant à distinguer des séquences musicales réelles des séquences générées artificiellement [2].

De façon classique, nous avons modélisé chaque séquence par un réseau de neurones récurrent de type *long-short term memory* (LSTM) avec une couche dense pour la classification finale en deux classes : « contient une fausse note » ou « ne contient pas de fausse note ». Chaque note a été représentée par la concaténation de trois vecteurs *one-hot* : un vecteur de taille 13 encodant sa hauteur (12 demi-tons et 1 silence), un vecteur de taille 5 encodant son octave (5 octaves) et un vecteur de taille 60 encodant sa durée (60 *tatums* par noire). Le réseau a été appris par le critère d'entropie croisée et la valeur de sortie est un score entre 0 et 1 indiquant la probabilité que la séquence ne contienne aucune fausse note.

Pour le premier jeu de données (*Omnibook*), nous avons obtenu une F-mesure moyenne de 82%, ce qui est bien mieux que le hasard mais vraisemblablement pas aussi bien qu'un auditeur humain. Pour le deuxième jeu de données (musique classique), nous avons obtenu une F-mesure moyenne de 52%, ce qui n'est guère mieux que le hasard. Nous n'avons pas encore étendu ce système à la prise en compte des autres musiciens. Ce travail est décrit en détail dans le rapport de stage de Rémi Decelle [1], qui est consultable en ligne.

[1] https://magenta.tensorflow.org/

Adaptation de la génération par apprentissage par renforcement

Supposons maintenant que nous disposons d'un système capable d'estimer un score entre 0 et 1 indiquant la probabilité de fausse note dans une séquence, prenant en compte le jeu des autres musiciens. Nous pouvons utiliser ce score pour adapter la génération. Pour cela, nous supposons que le système d'improvisation automatique et les musiciens sont représentés par des oracles des facteurs et qu'à chaque instant ils doivent chacun choisir parmi un nombre fini d'actions avec des probabilités connues (cf. Livrable 2.1.2). Pour chaque action possible du système d'improvisation automatique, il existe donc un nombre fini de combinaisons possibles d'actions des autres musiciens, chacune étant associée à une probabilité d'être jouée et à un score d'absence de fausse note. L'espérance du score sur l'ensemble des combinaisons est égale à la somme des scores pondérés par ces probabilités. Le système d'improvisation automatique choisit alors l'action dont l'espérance du score est la plus élevée. Cela crée une boucle entre la perception et l'action dans la mesure où l'action du système d'improvisation automatique influe sur le jeu des autres musiciens et donc sur sa perception.

Ce principe est celui de l'apprentissage par renforcement, qui constitue l'état de l'art dans le domaine de l'intelligence artificielle pour la modélisation des jeux (go, échecs…) mais aussi du dialogue homme-machine. Nous avons énoncé son usage pour l'adaptation de la génération musicale dans le rapport de stage de Rémi Decelle [1]. Mais, compte tenu des résultats décevants du système de détection de fausse note et du fait qu'il ne prend pas encore en compte le jeu des autres musiciens, nous ne l'avons pas évalué expérimentalement.

Modélisation par réseaux de neurones prenant en compte la structure du morceau

Après avoir observé les résultats décevants du système de détection de fausse note, nous avons consacré notre temps à améliorer la représentation des séquences musicales par réseaux de neurones récurrents. Les méthodes de l'état de l'art reposent sur l'hypothèse que la distribution conditionnelle d'une note sachant les notes précédentes est invariante par translation sur l'axe temporel. Dans la publication soumise [3] en annexe de ce rapport, nous proposons une nouvelle architecture de réseau de neurones récurrent à historique parallèle, où les paramètres du réseau de neurones dépendent de la position temporelle dans la séquence. Nous avons utilisé cette nouvelle architecture pour modéliser chaque bloc structurel [4] dans des morceaux constitués de blocs structurels de 16 mesures. Les expériences indiquent que la prise en compte de la non-invariance dans le temps permet de rendre compte de façon plus fine de la structure globale des mélodies apprises. Nous allons poursuivre nos recherches dans cette voie pendant les 18 mois restants de la thèse de Nathan Libermann. Il est à noter que, si ces travaux sont présentés ici, ils sont aussi liés au WP2.3 « Apprentissage de structures multi-échelles » dans la mesure où ils montrent que la prise en compte de la structure globale du morceau améliore la génération à une échelle locale. La modélisation de la structure globale (dans notre exemple, les liens entre les blocs de 16 mesures) par réseaux de neurones constitue une piste prometteuse pour le futur.

Références

[1] Rémi Decelle, "Apprentissage par renforcement pour l'improvisation musicale automatique", MSc thesis, Télécom Nancy, 2017. URL : https://hal.inria.fr/hal-01591521/document

[2] Li-Chia Yang, Szu-Yu Chou, Yi-Hsuan Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation", in *Proc. ISMIR*, pp. 324–331, 2017.

[3] Nathan Libermann, Frédéric Bimbot, Emmanuel Vincent, "Exploration de dépendances structurelles mélodiques par réseaux de neurones récurrents", soumis aux *Journées d'Informatique Musicale*, 2018.

[4] Frédéric Bimbot, Emmanuel Deruty, Gabriel Sargent, Emmanuel Vincent, "System & Contrast: A polymorphous model of the inner organization of structural segments within music pieces" *Music Perception*, 33(5): 631–661, 2016.

EXPLORATION DE DÉPENDANCES STRUCTURELLES MÉLODIQUES PAR RÉSEAUX DE NEURONES RÉCURRENTS

Nathan Libermann
Univ Rennes 1
nathan.libermann@inria.fr

Frédéric Bimbot
IRISA/CNRS
frederic.bimbot@irisa.fr

Emmanuel Vincent
INRIA
emmanuel.vincent@inria.fr

RÉSUMÉ

Dans le cadre de la génération automatique de mélodie structurée, nous explorons la question des dépendances entre les notes d'une mélodie en utilisant des outils d'apprentissage profond. Plus précisément, nous utilisons le modèle d'apprentissage séquentiel GRU, que nous déclinons dans différents scénarios d'apprentissage afin de mieux comprendre les architectures optimales dans ce contexte. Pour tenir compte de la non-invariance temporelle des dépendances entre les notes au sein d'un segment structurel (motif, phrase, section) nous souhaitons par ce moyen explorer différentes hypothèses. Nous définissons trois types d'architectures récurrentes correspondant à différents schémas d'exploitation de l'historique musical dont nous étudions les capacités d'encodage et de généralisation. Ces expériences sont conduites sur la base de données Lakh MIDI Dataset et plus particulièrement sur un sous-ensemble de 8308 segments mélodiques monophoniques composés de 16 mesures. Les résultats indiquent une distribution non-uniforme des capacités de modélisation et de prédiction des réseaux récurrents testés, suggérant l'utilité d'un modèle non-ergodique segments mélodiques.

1. INTRODUCTION

La génération automatique de mélodie est une problématique régulièrement abordée en informatique musicale mais qui reste incomplètement résolu. Récemment revenues au premier plan, les méthodes par réseaux de neurones apparaissent potentiellement capables de modéliser des mécanismes de génération de mélodie par apprentissage à partir d'exemples.

Chen et al [2] furent parmis les premiers auteur à publier sur ce sujet. L'un des principaux problèmes qu'ils relèvent est le manque de structure globale dans les mélodies générées. D'autres auteurs comme Franklin [3], Eck et Schmidhuber [4] ont alors cherché à résoudre ce problème en utilisant des réseaux récurrent LSTM [5] sur un corpus de musique blues ou jazz. Ces travaux ont permis d'améliorer la qualité perçue de la musique générée mais les résultats continuent à présenter une insuffisance de structure. Boulanger-Lewandowski et al. [6] ont tenté de combiner le modèle LSTM avec des modèles génératifs, notamment le RBM [7]. Dans Huang et Wu [8], les auteurs s'intéressent à la représentation des notes dans un espace vectoriel (embedding). Jaques et al [9] proposent de restreindre un modèle LSTM préalablement appris grâce à l'apprentissage par renforcement de règles de musicologie prédéfinies. L'équipe Magenta [1] propose un modèle d'attention inspiré de Bahdanau et al [10]. A notre connaissance, il n'existe pas aujourd'hui de modèle capable d'apprendre à générer des mélodies présentant une structure pleinement satisfaisante à l'échelle de plusieurs mesures consécutives.

Le travail présenté dans cet article est une étude exploratoire qui s'inscrit dans ce cadre de la génération automatique de mélodie structurée, et qui fait appel à des outils d'apprentissage profond. Plus particulièrement nous considérons le modèle séquentiel GRU (Gated Recurrent Units) [11], que nous étudions dans différents scénarios d'apprentissage afin de mieux comprendre les potentialités de cette approche pour la modélisation de mélodies. Nous souhaitons notamment cerner l'importance d'une hypothèse de non-ergodicité (non-invariance dans le temps) de la structure musicale, en mettant en évidence les limites des architectures récurrentes à base de GRU et étudier les possibilités de les adapter à la génération de motifs mélodiques. On suppose en effet qu'il existe une planification dans la construction d'un segment mélodique qui ne se contente pas de se référer aux k précédents éléments pour construire le suivant. Au contraire, nous faisont l'hypothèse qu'un segment mélodique forme un tout et qu'au fil du segment les divers éléments se conditionnent de façon non-adjacente pour former le schéma mélodique global.

Ainsi, dans l'esprit des travaux récents sur les modèles tensoriels/polytopiques de segments musicaux [12] [13], nous émettons l'hypothèse que dans le cadre de mélodies simples constituées de motifs présentant des relations d'analogie, les dépendances structurelles dans la musique ne suivent pas un procédé purement séquentiel, mais plutôt des dépendances multi-échelles. Selon cette approche, un élément musical dépend de façon privilégiée des autres éléments qui se situent dans des positions métriques homologues dans le segment plutôt que dans le voisinage immédiat. Autrement dit, les positions métriques des notes jouent un rôle dans la construction structurelle de la musique et les architectures neuronales doivent en tenir compte.

Nous définissons donc trois modèles récurrents à base de GRU. Un modèle à historique glissant, qui correspond à une façon "standard" de construire et d'entrainer un mo-

1 . https ://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn

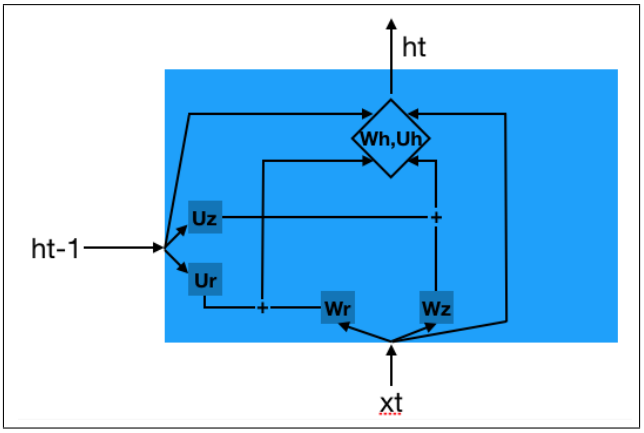


Figure 1. GRU

dèle récurrent. Un modèle à historique croissant, qui correspond à une façon "dynamique" d'apprendre un modèle récurrent. Et enfin, un modèle à historique parallèle, avec des poids distincts selon les positions à prédire.

La comparaison des différents schémas d'apprentissage proposé selon leur capacité d'encodage de l'information musicale et de leur performance de prédiction permet d'étudier la pertinence et les limites de l'hypothèse de non-ergodicité dans les séquences mélodiques.

2. PROTOCOLE EXPÉRIMENTAL

2.1. Cellule de mémoire GRU et couche de prédiction

Pour définir les architectures étudiées, nous utilisons comme unité de base le réseau de neurones récurrent GRU (*Gated Recurrent Unit*, voir Fig. 1) qui fonctionne comme suit. A chaque instant t la cellule GRU reçoit en entrée, sous forme de vecteurs, l'observation courante x_t et une variable interne h_{t-1} qui tient lieu de mémoire de l'historique des observations précédentes. A partir de ces deux entrées, la cellule GRU produit une remise à jour de h, laquelle est ensuite utilisée dans une cellule GRU semblable, prenant h_t et l'observation x_{t+1} en entrée,... et ainsi de suite. Dans nos expériences, les vecteurs x_t correspondant à un vecteur one-hot sur un ensemble discret de notes (voir section 2.3).

La cellule GRU se décompose en 6 sous-ensembles de poids de *propagation d'historique* (U_h, U_r, U_z, W_h, W_r, W_z) qui se combinent avec les entrées h_{t-1} et x_t pour former h_t selon les équations suivantes :

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (1)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (2)$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \quad (3)$$

où $\circ$ désigne le produit matriciel de Hadamard.

Nous avons aussi besoin pour définir nos architectures d'une couche de prédiction qui à partir d'une mémoire h_t fournit une prédiction $\_x_{t+1}$ de la note suivante x_{t+1}.

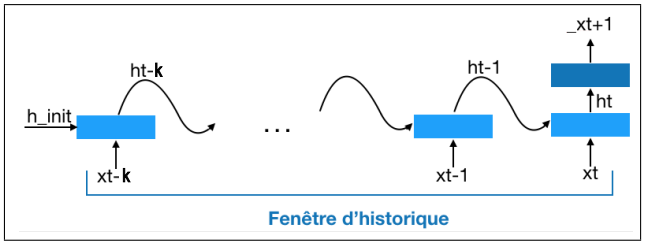


Figure 2. Architecture à historique glissant

Pour ce faire, nous utilisons une couche entièrement connectée entre h_t et $\_x_{t+1}$, constituée d'une matrice de poids de *prédiction* W_p et d'un vecteur de biais b_p, ce qui fournit en sortie une distribution de probabilité a posteriori sur l'ensemble des notes :

$$\_x_{t+1} = W_p h_t + b_p \quad (4)$$

2.2. Spécification des architectures

Dans ce travail nous considérons des segments mélodiques correspondant à des unités structurelles de type phrase ou section musicale. Dans cette optique, nous modélisons des séquences de $N = 64$ notes qui sont obtenues en échantillonnant des mélodies qui s'étendent sur 16 mesures.

Trois architectures à base de GRUs sont étudiées :

— L'architecture à *historique glissant* correspond à une façon standard de construire un réseau récurrent. On choisit une fenêtre d'historique de taille fixe k. Pour prédire la note $\_x_{t+1}$, on alimente la couche de prédiction W_p avec l'historique h_t, qui, dans ce cas, est initialisé aléatoirement puis propagé depuis h_{t-k} jusqu'à h_t (voir Figure 2). La fenêtre d'historique $[t-k, t]$ étant fixe, on ne peut calculer les prédictions $\_x_{t+1}$ qu'à partir de l'instant $k+1$. Les poids des couches de propagation d'historique sont partagés (c'est-à-dire indépendants de t) et il en est de même des poids des couches de prédiction. Ainsi, cette architecture repose sur une hypothèse d'invariance dans le temps des éléments musicaux.

— L'architecture à *historique croissant* correspond à une façon plus dynamique de construire un modèle récurrent. Ici on procède de la même façon que pour l'architecture du modèle à historique glissant, mais on considère une fenêtre d'historique qui couvre l'intégralité de l'intervalle $[1, t]$ et qui par conséquent croit au fur et à mesure que l'on progresse dans le temps (voir Figure 3). Comme dans l'architecture précédente, les poids de la couche de propagation d'historique sont communs pour les différentes positions de notes à prédire et il en est de même pour les poids de la couche de prédiction. Mais dans cette variante, il devient possible de prédire les notes dans toutes les positions temporelles, à partir d'un historique qui va croissant. On note toutefois que cette architecture repose aussi

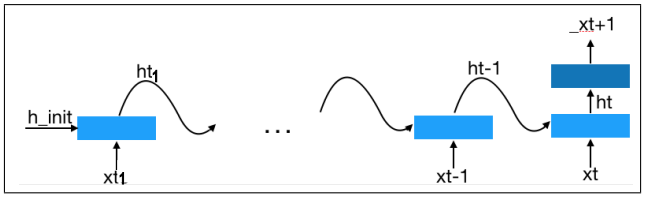


Figure 3. Architecture à historique croissant

sur une hypothèse d'invariance dans le temps du fait du partage des poids.

— Enfin, l'architecture à *historique parallèle* repose sur le même principe que l'architecture à historique croissant, à ceci près que les poids des couches de propagation et de prédiction sont indépendants pour chaque position à prédire $\_x_{t+1}$. On peut ainsi voir cette configuration comme $N-1$ réseaux à historique croissant, indépendants les uns des autres et correspondant à chaque position à prédire $\_x_{t+1}$. L'indépendance de ces réseaux en fonction de la position est censée permettre à cette architecture de prendre en compte la non-invariance dans le temps et de tenir compte, si nécessaire, de dépendances complexes.

2.3. Données

Pour ce travail d'exploration, nous utilisons les données du Lakh MIDI Dataset [1], qui contient 176581 fichiers MIDI multipistes. Etant donné que nous nous intéressons uniquement aux pistes mélodiques monophoniques, nous en avons sélectionné environ 70000 qui possédaient cette propriété dans le corpus originel. De ce sous-ensemble, nous avons extrait 8308 blocs structurels de 16 mesures sous forme de fichier MIDI et correspondant aux 16 premières mesures de mélodies monophoniques de type 4/4.

Afin de permettre au modèle de pouvoir mieux extraire les relations relatives entre les notes, nous avons représenté ces mélodies en référence à la première note de la séquence (arbitrairement fixée à Do), de sorte à être indépendant de la tonalité initiale. Nous obtenons donc 8308 séquences mélodiques de 16 mesures.

Pour simplifier la représentation traitée, on considère dans un premier temps une discrétisation de l'information mélodique dans ces segments. Nous avons procédé à un découpage des mesures en 4 portions égales et relevé à chaque fois la note active sur ces positions. Si un silence apparaît sur l'un des temps considérés, nous prolongeons la valeur de la note précédente ce qui évite d'avoir à gérer des absences de notes.

Le résultat de ce processus de réduction conduit à des séquences "mélodiques" de 16 mesures correspondent toutes à des successions de 64 notes. Chaque note est représentée par un vecteur de dimension 88 (correspondant aux 88 notes d'un clavier de piano). Lorsqu'une note est active, la dimension associée à cette note dans le vecteur est mise à 1 et toutes les autres à 0.

Train	Test	Test
A	A	B
B	B	A

Table 1. Protocoles d'apprentissage et de test pour mesurer les capacités de compression et de généralisation des architectures

2.4. Protocole

Le corpus composé de 8308 séquences est divisé en deux groupes A et B comprenant chacun 4154 séquences. Nous définissons quatre scénarios pour chaque architecture voir Table 1 :

— Apprentissage sur le groupe A, test sur le groupe A
— Apprentissage sur le groupe A, test sur le groupe B
— Apprentissage sur le groupe B, test sur le groupe A
— Apprentissage sur le groupe B, test sur le groupe B

Lors du test, nous relevons pour chaque positions de note, l'erreur moyenne sur l'ensemble des exemples de test, ceci dans le cadre des 3 architectures décrites dans la section 2.2.

Les mesures effectuées en utilisant le même ensemble de test que celui utilisé pour l'apprentissage permettent de caractériser les capacités de *compression* de l'information mélodique par les différentes architectures, en fonction de la position de la note dans le segment musical. Celles qui croisent les ensembles d'apprentissage et de test rendent compte la capacité de *généralisation* des architectures à des données nouvelles.

2.5. Critère d'évaluation

Pour chaque instant t, nous calculons l'erreur entre le vecteur de sortie $\_x_t$ et la note réelle x_t par la fonction d'erreur quadratique moyenne (MSE). On rappelle que $\_x$ est un vecteur de dimension $n = 8$ correspondant à une distribution de probabilités et x un vecteur one-hot de même dimension, correspondant à la note effectivement active.

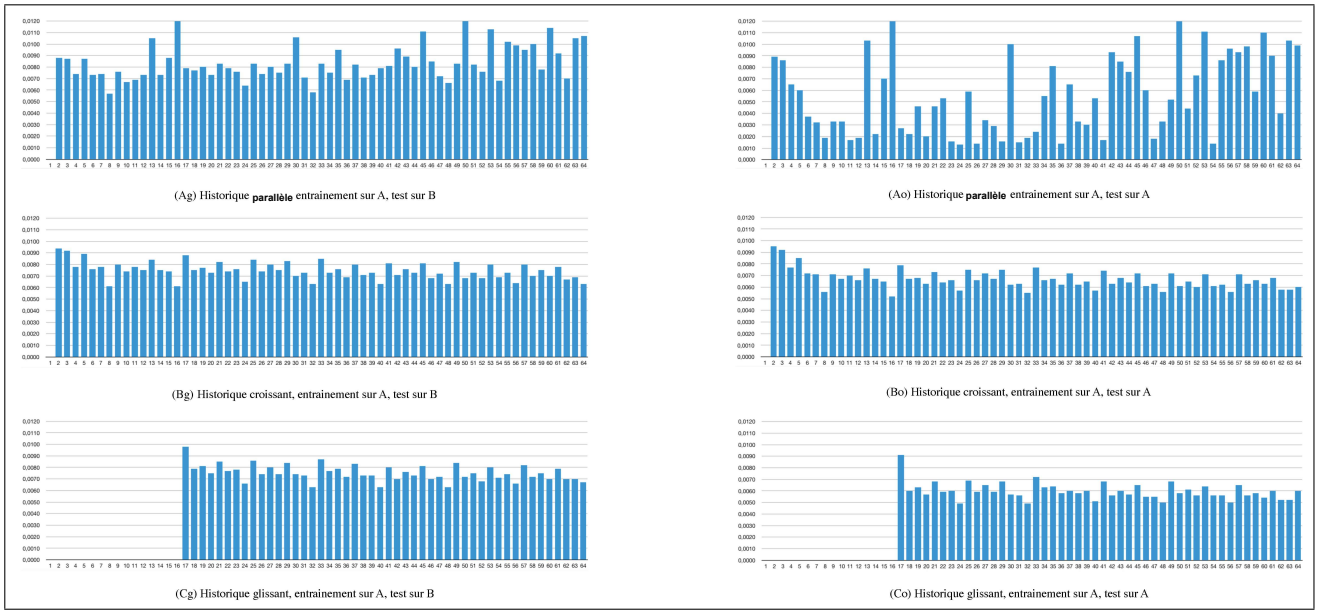
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\_x_i - x_i)^2 \qquad (5)$$

C'est l'évolution de cette erreur que nous examinons pour chaque position de note et dans chaque scénario, afin d'observer la façon dont les différentes architectures encodent les dépendances structurelles. Nous analysons à la fois les motifs engendrés par la variation de l'erreur moyenne entre les différentes positions ainsi que le critère de compression / prédiction.

2.6. Détails complémentaires de mise en oeuvre

Pour ces expériences nous utilisons la bibliothèque d'apprentissage profond Pytorch [2] . L'historique est un vecteur de dimension 100. Chaque exemple est présenté 40 fois au

2 . http ://www.pytorch.org



(Ag) Historique **parallèle** entrainement sur A, test sur B

(Ao) Historique **parallèle** entrainement sur A, test sur A

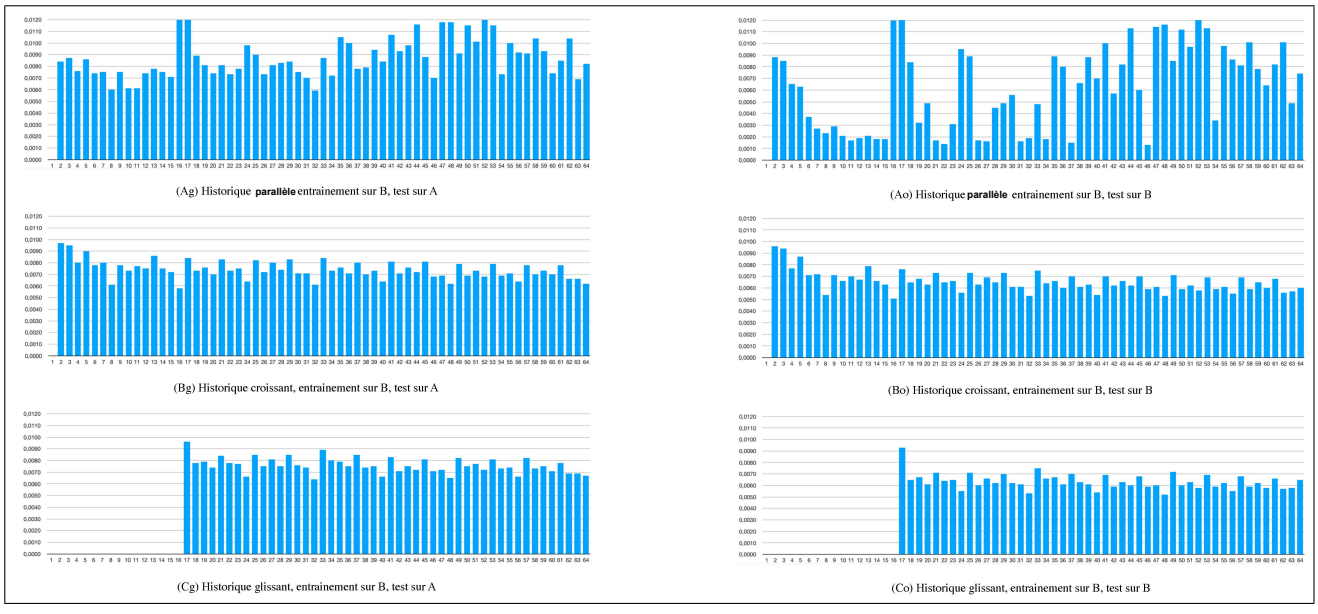
(Bg) Historique croissant, entrainement sur A, test sur B

(Bo) Historique croissant, entrainement sur A, test sur A

(Cg) Historique glissant, entrainement sur A, test sur B

(Co) Historique glissant, entrainement sur A, test sur A

Figure 4.



(Ag) Historique **parallèle** entrainement sur B, test sur A

(Ao) Historique **parallèle** entrainement sur B, test sur B

(Bg) Historique croissant, entrainement sur B, test sur A

(Bo) Historique croissant, entrainement sur B, test sur B

(Cg) Historique glissant, entrainement sur B, test sur A

(Co) Historique glissant, entrainement sur B, test sur B

Figure 5.

cours de l'apprentissage (40 epochs). L'optimiseur utilisé est Adagrad.

3. RÉSULTAT

Les figures 4 et 5 illustrent les résultats du protocole expérimental décrit dans la section précédente. La figure 4 correspond aux différentes architectures apprises à partir du corpus A et la figure 5 aux architectures apprises sur le corpus B.

Pour chacune de ces figures, les résultats de généralisation des architectures sont représentés sur les graphiques Ag, Bg, Cg et les résultats de compression sur les graphiques Ao, Bo et Co.

— graphique A : architecture à historique parallèle
— graphique B : architecture à historique croissant
— graphique C : architecture à historique glissant

En premier lieu, on constate une similarité de comportement entre courbes homologues sur la figure 4 et la figure 5. Ceci permet de penser que les tendances observées sont relativement peu influencées par les spécificités des deux demi-corpus.

On remarque ensuite que pour les architectures à historique glissant et croissant, les valeurs des capacités de généralisation (Bg et Cg) présentent un niveau d'erreur à peine plus élévé que les valeurs des capacités de compression (Bo et Co). A l'inverse, les capacités de compression et de généralisation de l'architecture parallèle se comportent très différemment l'une de l'autre. En effet, l'architecture parallèle possède beaucoup plus de paramètres libres ce qui crée une capacité bien plus forte du réseau à comprimer les données d'apprentissage mais entraîne un phénomène d'over-fitting (surapprentissage) qui altère la capacité de généralisation sur de nouvelle données. Ce phénomène est bien moins saillant sur les deux autres architectures.

Un examen plus précis des motifs observés sur les courbes d'erreur apporte également des observations intéressantes.

Les courbes pour les historiques glissant et croissant présentent une pseudo-periodicité marquée à l'échelle de 8 notes et des oscillations secondaires aux échelles 4 et 2, suggérant fortement une "synchronisation" des capacités de modélisation des architectures correspondantes sur des cycles de 2 mesures.

Les courbes Ao (que ce soit pour la figure 4 ou la figure 5) présentent pour leur part un comportement plus contrasté : tout d'abord une décroissance très nette sur le premier quart du segment puis des variations plus erratiques sur la partie centrale (2ème et 3ème quart) et enfin une remontée globale de l'erreur sur le dernier quart. On peut opérer un rapprochement entre ces observations expérimentales et différentes considérations musicologiques sur la structure des segments musicaux : notamment que l'on constate souvent la réalisation de formules musicales conventionnelles en fin de segment, donc finalement moins prédictibles en fonction du contexte, puisqu'elle sont plus ou moins prédéterminées d'avance indépendemment de celui-ci.

Par ailleurs, selon le modèle cognitif d'Implication-Réalisation de Narmour [14] (et son extension récente [15]), les fins de segments structurels constituent fréquemment des dénis d'implication par rapport aux progression musicales établies dans les portions antérieures, ce qui peut également constituer une hypothèse expliquant le comportement observé sur les courbes Ao. Toutefois ce comportement étant moins net sur les courbes Ag, des expériences complémentaires sont requises pour mieux asseoir ces hypothèses.

4. CONCLUSIONS

Dans ce travail d'exploration, nous avons considéré trois architectures de réseaux de neurones récurrents et nous avons analysé leur comportement en terme d'erreur de modélisation de schémas mélodiques simplifiés.

Ces expériences nous ont permis d'observer que la prise en compte de la non-invariance dans le temps de la structure musicale dans une architecture de réseau de neurones récurrent permet de rendre compte de façon plus fine de la structure globale des mélodies apprises par le réseau.

Dans le cadre de nos expériences actuelles, cette conclusion demeure partielle, du fait d'un volume de données d'apprentissage limité qui entraîne une capacité insuffisante de généralisation du réseau appris.

Toutefois, cette première exploration de l'hypothèse de non-invariance dans le temps de la structure musicale par des réseaux de neurones récurrents nous conforte dans l'idée de poursuivre nos recherches dans cette voie.

Plusieurs pistes sont prévues pour compléter ces travaux à court terme, permettant ainsi d'enrichir ces premières investigations par des expériences supplémentaires et les résultats correspondants.

5. REFERENCES

[1] Raffel, C. "Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matchingr", *PhD Thesis*, Colombia, USA, 2016.

[2] Chen, J., Miikkulainen, R. "Creating Melodies with Evolving Recurrent Neural Networks", *Proceedings of the 2001 International Joint Conference on Neural Networks. IEEE*, Washington, USA, 2001.

[3] Franklin, J. "Jazz Melody Generation from Recurrent Network Learning of Several Human Melodies" *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, Florida, USA, 2005.

[4] Eck, D. Schmidhuber, J. "Finding temporal structure in music : Blues improvisation with LSTM recurrent networks" *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, 2002.

[5] Hochreiter, S. Schmidhuber, J. "Long Short-Term memory" *Neural Computation*, 1997.

[6] Boulanger-Lewandowski, N. Bengio, Y. Vincent, P. "Modeling Temporal Dependencies in High-Dimensional Sequences : Application to Polyphonic Music Generation and Transcription" , 2012.

[7] Smolensky, P. "Information processing in dynamical systems : foundations of harmony theory" *MIT Press Cambridge*, MA, USA, 1986.

[8] Huang, A. Wu, R. "Deep Learning for Music" , 2016.

[9] Jaques, N. Gu, S. Turner, R. Eck D. "Generating Music by Fine-Tuning Recurrent Neural Networks with Reinforcement Learning" ,.

[10] Bahdanau, D. Cho, K. Bengio, Y. "Neural Machine Translation by Jointly Learning to Align and Translate" , 2016.

[11] Cho, K. Van Merriënboer, B. Gulcehre, C. Bahdanau, D. Bougares, F. Schwenk, H. Bengio, Y. "Learning phrase representations using RNN encoder-decoder for statistical machine translation" , 2014

[12] Guichaoua, C. "Modèles de compression et critères de complexité pour la description et l'inférence de structure musicale" Thèse supervisé par Bimbot, F., France, 2017

[13] Louboutin, C. Bimbot, F. "Description of chord progressions by minimal transport graphs using the System Contrast model" *proceedings of the 42nd International Computer Music Conference*, 2016

[14] Narmour, E. "The analysis and cognition of basic melodic structures : the implication-realization model" *Univ. of Chicago Press.*, USA, 1990

[15] Bimbot, F. Deruty, E. Sargent, G. Vincent, E. "System Contrast : A Polymorphous Model of the Inner Organization of Structural Segments within Music Pieces" *Music Perception, University of California Press*, USA, 2016