Université de Bordeaux

Rapport de stage
Recherche sur la synthèse sonore/musicale à partir d'approches basées sur l'apprentissage profond

Auteur :
Nicolas ETCHEVERRY

Tuteurs de Stage :
Pierre Hanna
Myriam Desainte-Catherine
Enseignant référent :
Pierre Bénard

3 septembre 2018

Résumé

L'objectif de ce stage est d'étudier différentes méthodes de synthèse et de traitement du son qui utilisent des réseaux de neurones profonds dans le but d'appliquer à un son une caractéristique particulière d'un autre son. Une méthode est retenue pour effectuer une expérience : le transfert de style. Cette méthode implique d'extraire une information de style d'une couche d'un réseau de neurones convolutionnel. Ces informations de style peuvent être transférées à un autre son par descente de gradient. Deux méthodes différentes sont implémentées : un AlexNet avec entrainement sur des données brutes et un VGG19 pré-entainé sur des images dans lequel on fait passer des spectrogrammes. Le résultat est que la méthode de l'AlexNet est plus cohérente que VGG19, mais VGG19 produit quand même des sons audibles sans qu'on puisse expliquer ce qui a été transféré.

The goal of this internship is to study several audio synthesis and processing methods based on Deep Neural Networks in order to apply to a sound a specific characteristic from another sound. An experiment is conducted on style transfer. This method consists of the extraction of style information from a convolutionnal network layer. This style information can be transfered to another sound by gradient descent. Two methods have been implemented : AlexNet with temporal data input and VGG19 with image recognition pre-training with spectrograms as input. The result is that AlexNet seems more coherent than VGG19, but VGG19 still produces audible sounds without us being able to explain which part was transferred.

Table des matières

1 Contexte **1**
- 1.1 SCRIME . . . 1
- 1.2 Sujet . . . 1
- 1.3 Organisation . . . 1

2 Etat de l'art **2**

3 Travail réalisé **4**
- 3.1 Apprentissage profond . . . 4
  - 3.1.1 Réseau de neurones profond . . . 4
  - 3.1.2 Réseau de neurones convolutionnel . . . 5
  - 3.1.3 Keras, Tensorflow, Theano . . . 5
- 3.2 Alexnet . . . 6
  - 3.2.1 Méthode . . . 6
  - 3.2.2 Architecture du réseau . . . 6
  - 3.2.3 Données d'entrée . . . 6
  - 3.2.4 Entrainement . . . 7
  - 3.2.5 Extraction du style . . . 7
  - 3.2.6 Transfert de style . . . 7
  - 3.2.7 Résultats . . . 8
    - 3.2.7.1 Qualité des classifieurs . . . 8
    - 3.2.7.2 Transfert de style . . . 8
    - 3.2.7.3 Etude du style . . . 8
- 3.3 VGG19 . . . 11
  - 3.3.1 Méthode . . . 11
  - 3.3.2 Architecture . . . 11
  - 3.3.3 Entrée . . . 11
  - 3.3.4 Entrainement . . . 12
  - 3.3.5 Transfert de style . . . 12
  - 3.3.6 Visualisation . . . 12
- 3.4 Résultats . . . 13
  - 3.4.1 Version pré-entraînée sur des images . . . 13
    - 3.4.1.1 Identité . . . 13
    - 3.4.1.2 Résultats intéressants à l'écoute . . . 13
    - 3.4.1.3 Bruit blanc . . . 14
    - 3.4.1.4 Silence . . . 15
    - 3.4.1.5 Pics sinusoïdaux . . . 15
    - 3.4.1.6 Prétraitement d'intensité . . . 16
    - 3.4.1.7 Etude du style . . . 16

4 Bilan **19**

5 Perspectives **20**

Chapitre 1

Contexte

1.1 SCRIME

Le Studio de Création et de Recherche en Informatique et Musiques Expérimentales (SCRIME) est un centre d'activités administré par le LaBRI. Il abrite essentiellement des projets liés au son et à la musique mettant en relation aussi bien des scientifiques que des artistes.

1.2 Sujet

L'objectif du stage est d'étudier les différentes méthodes de synthèse et de traitement de sons numériques utilisant des réseaux de neurones profonds. L'application recherchée est de pouvoir modifier un son pour lui donner une caractéristique particulière d'un autre son. Après avoir pris connaissance de l'existant, le sujet s'est précisé en une implémentation et analyse de différentes méthodes de transformation de sons par transfert de style.

1.3 Organisation

Le stage s'est déroulé dans un bureau du bâtiment A37 (Château Bonnefont) partagé avec deux autres stagiaires de Master 2. Une réunion avec mes tuteurs de stage Pierre Hanna et Myriam Desainte-Catherine avait lieu une fois par semaine environ pour faire un point sur les avancées et définir les actions de la semaine suivante.

Mon ordinateur personnel n'étant pas suffisamment puissant pour exécuter des fonctions de réseaux de neurones, l'essentiel de mon code a été écrit et exécuté à distance au CREMI par SSH en bureau distant sur le serveur Tesla (multi GPU). Par nature, les réseaux de neurones prennent beaucoup de temps à s'entrainer et à s'exécuter. Pour référence, les entrainement peuvent durer jusqu'à une semaine et la production d'un son prend environ 3h.

Tous les sons et les images générés lors de ce stage sont disponibles ici : `http://nicho.fr/transferotron`
Le code source est disponible ici : `https://github.com/etcheverry/transferotron`

Chapitre 2

Etat de l'art

Dans ce chapitre, nous nous intéresserons aux différentes méthodes de synthèse et de transformation de sons basées sur de l'apprentissage profond.

WaveNet [11] est un réseau de neurones convolutionnel permettant de générer de l'audio brut. Il est surtout utilisé pour la synthèse vocale en "text to speech" et fournit de très bons résultats dans ce domaine. Ces réseaux sont aussi capables de générer de la musique [3], notamment du piano, si on l'entraine avec la base de données de piano solo de YouTube.

Basé sur WaveNet, le synthétiseur NSynth [3] de Magenta (Google) a pour objectif de factoriser la musique en dissociant la note de musique d'un son de ses caractéristiques audio. Ainsi, il peut générer des sons combinant plusieurs caractéristiques audio venant d'instruments différents. Il est aussi possible de spécifier la note à laquelle on veut générer le son. La base de donnée "NSynth", évoquée dans le même article que le synthétiseur, est une base de données de 305 979 notes de musique venant de 1006 instruments différents, annotés avec diverses caractéristiques des sons comme la façon de jouer, la note, la famille d'instrument ou la qualité de l'enregistrement.

Les GANs (Generative Adversarial Network) [6] sont un système combinant deux réseaux de neurones profonds : un générateur et un discriminateur. Le générateur crée des éléments et le discriminateur est entrainé à déterminer si un élément est réel ou non car on lui fournit un ensemble d'éléments réels en plus de l'élément généré. On entraine le générateur à tromper le discriminateur et celui-ci finira par générer des éléments qui semblent "réels". Les GANs ont prouvé leur efficacité dans le domaine de l'image en générant des images à l'apparence réaliste.

Chrise Donahue, Julian McAuley et Miller Puckette [2] proposent une méthode inspirée du GAN baptisée WaveGAN qui utilise directement la représentation temporelle des sons, plutôt que d'utiliser les spectrogrammes et d'appliquer naïvement la méthode classique des GANs.

Leon Gatys, Alexander Ecker et Matthias Bethge [4] proposent une méthode permettant d'extraire et de transférer le style d'une image pour le transférer sur une autre en conservant son contenu. L'algorithme est le suivant : un réseau de neurones convolutionnel est entrainé à reconnaitre des classes dans une image. On extrait les informations intermédiaires contenues dans les couches de convolution avec lesquelles on construit une matrice de style et une matrice de contenu pour deux images : une image qui fournirai le style et une qui fournira le contenu. On applique ensuite une descente de gradient sur une image initialisée pour minimiser la différence entre sa matrice de style et celle de l'image de style, ainsi que la différence entre sa matrice de contenu et celle de l'image de contenu. Des exemples montrent comment cet algorithme transfère plusieurs styles de dessin sur des photographies d'une ville, sans abimer les formes des habitations.

Inspirés par ces résultats, plusieurs applications de cette méthode ont vu le jour dans le domaine du son.

Cependant, s'il est facile de reconnaitre le "style" d'une image de façon subjective, c'est plus compliqué quand il s'agit du son. Shuqi Dai, Zheng Zhang et Gus Guangyu Xia [1] proposent de séparer ces notions en trois catégories. La première forme de "Style" serait l'organisation d'un morceau et sa partition, la deuxième serait le contrôle de la performance (puissance de jeu, dynamique, durée, décalages...) et la troisième serait les caractéristiques du son brut, c'est-à-dire son timbre. Dans la suite de l'étude, nous allons nous concentrer sur le transfert de timbre.

Prateek Verma et Julius O. Smith [12], inspirés par la méthode de transfert de style d'image, proposent une méthode permettant de créer de nouveaux sons avec la même approche. Ils utilisent des spectrogrammes en entrée et les architectures AlexNet, VGG-Net et ResNet pour combiner le style d'un son avec le contenu d'un autre son. Les résultats semblent satisfaisants.

Eric Grinstein, Ngoc Duong, Alexey Ozerov et Partick Pérez [7] proposent une approche similaire mais utilisent des réseaux différents : VGG19, SoundNet et un réseau de neurones initialisé aléatoirement. Ils présentent également une approche basée sur le modèle de McDermott, un réseau conçu pour imiter le système auditif humain. Leur résultats sont plutôt satisfaisants. Ils concluent également de leurs expérimentations qu'initialiser la descente de gradient avec le son de contenu et ne plus se soucier de la matrice de contenu permet d'obtenir de meilleurs résultats.

Chapitre 3

Travail réalisé

3.1 Apprentissage profond

La première partie de ce stage a consisté pour moi à comprendre comment fonctionnait un réseau de neurones, et un réseau de neurones convolutionnel.

3.1.1 Réseau de neurones profond

Un réseau de neurones profond est un graphe orienté dans lequel les sommets sont organisés en "couches". Chaque sommet, ou "neurone", d'une couche est relié à l'ensemble des neurones de la couche suivante. La première couche est la couche d'entrée, la dernière est la couche de sortie et les couches intermédiaires sont les couches cachées.

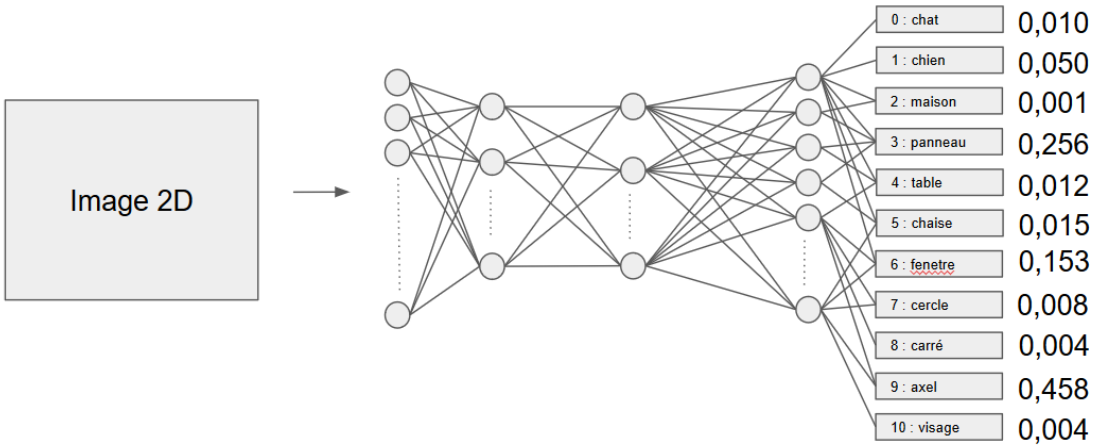

FIGURE 3.1 – Réseau de neurones profond

Chaque arête a un poids, et chaque sommet possède une "valeur d'activation" dépendant du poids des arêtes entrantes et des valeurs d'activations des sommets précédents.

Ainsi, un ensemble de données d'entrées va parcourir le graphe et être altéré par les poids des arêtes jusqu'à atteindre la sortie. Les valeurs de sortie du réseau sont généralement comprises entre 0 et 1 et représentent la probabilité que la donnée d'entrée appartienne à cette classe.

L'entraînement d'un réseau de neurones profond consiste à passer des entrées dont on connaît la sortie attendue et à modifier les poids des arêtes jusqu'à ce que chaque entrée retourne la sortie correcte (dans l'idéal).

Pour ce faire, on applique un algorithme de "back-propagation" qui permet d'optimiser le choix des modifications à faire sur les poids en fonction du gradient entre les différentes couches.

3.1.2 Réseau de neurones convolutionnel

Un réseau de neurones convolutionnel fonctionne sur le même principe que le réseau précédent mais dispose de plusieurs types de couches supplémentaires.

Les couches de convolution vont appliquer un algorithme de convolution sur l'image qui la traverse à l'aide d'un filtre. Ce sont les valeurs de ce filtre qui vont déterminer l'opération que la convolution va effectuer, et ces valeurs sont définies lors de l'entrainement du réseau. Une couche de convolution peut avoir un ou plusieurs filtres, le nombre de filtre correspondant à la profondeur de la couche.

Des couches de "Pooling" sont également utilisées pour réduire la taille des images et rendre les représentations plus légères pour réduire le nombre de paramètres à entrainer. Ces couches ont pour résultat une sorte de "pixelisation" de l'image.

L'architecture d'un réseau de neurones convolutionnel est souvent de plusieurs couches de convolution suivies de couches denses, similaires à celles d'un réseau de neurones profond classique.

Ces réseaux de neurones font en théorie de bons classifieurs, plus rapides à entrainer qu'un réseau de neurones profonds classique.

3.1.3 Keras, Tensorflow, Theano

Keras est une bibliothèque python haut niveau de réseaux de neurones. Elle permet de choisir son back-end entre Tensorflow, Theano et CNTK, des bibliothèques bas niveau de réseaux de neurones. Au départ, j'utilisais Tensorflow mais il s'est avéré que certaines fonctions essentielles (comme le chargement de poids personnalisés sur un modèle) n'étaient pas implémentées. J'utiliserai donc Théano en back-end sur les programmes qui suivront. Le plus gros avantage de Keras est qu'elle permet de prototyper assez rapidement des modeles de réseaux, avec des fonctions plus simples à utiliser que celles de Tensorflow ou Theano.

3.2 Alexnet

3.2.1 Méthode

Dans cette partie, l'objectif est de reproduire l'expérience de P. Verma et J. O. Smith [12] qui proposent une méthode de transfert de style de son brut.

La méthode consiste à entraîner un réseau de neurones convolutionnel à reconnaître des instruments. Ensuite, pour transférer le style d'un son A sur un son B, on les passe d'abord dans le réseau puis on extrait les activations d'une couche de convolution du réseau pour calculer une matrice de style.

Ces matrices permettent de calculer une fonction de perte que l'on va minimiser par descente de gradient sur le son à modifier.

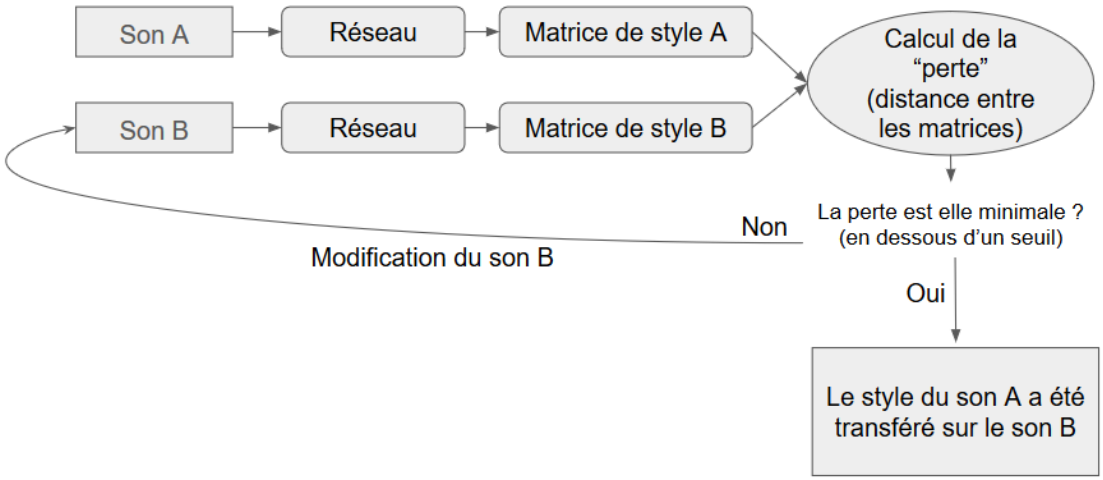

FIGURE 3.2 – Explication

L'intuition de cette méthode est que les informations extraites du réseau correspondent à une information abstraite caractéristique de la classe dans laquelle le son a été classé. C'est cette information qu'on va copier sur le son cible pour modifier son "style".

3.2.2 Architecture du réseau

AlexNet est un réseau de neurones convolutionnel comprenant 5 couches de convolution suivies de deux couches denses.

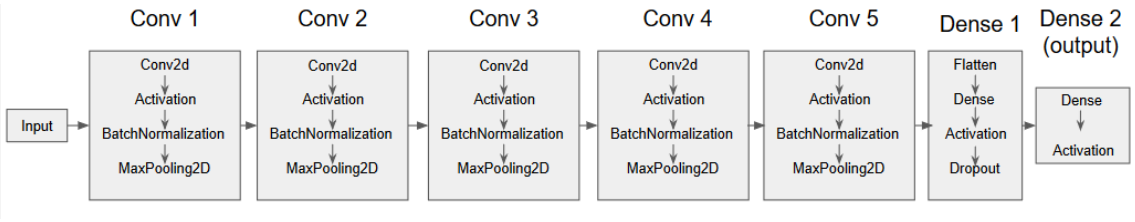

FIGURE 3.3 – Architecture AlexNet

Le réseau sera entrainé à classifier les sons en fonction de leur famille d'instruments.

3.2.3 Données d'entrée

Les données d'entrées sont issues de la banque de sons NSynth de Google. NSynth contient 305 979 enregistrements de 4 secondes d'instruments à différentes hauteurs. La banque est séparée en 3 ensembles : un ensemble

d'entrainement de 289 205 fichiers, un ensemble de validation de 12 678 fichiers et un ensemble de test de 4096 fichiers.

L'intéret de cette banque de données est qu'elle propose un fichier "json" contenant des caractéristiques très précises sur chaque enregistrement : la note jouée, l'instrument, la hauteur en MIDI, la vitesse d'impact en MIDI, la famille de l'instrument ou la source de l'instrument (accoustique, electronique ou synthétique).

La méthode de P. Verma et J. O. Smith [12] propose d'appliquer une transformée de fourier à court terme ("STFT" en anglais) sur les sons d'entrée pour obtenir leurs spectrogrammes d'amplitude. Cependant, les réseaux de neurones proposant habituellement des classifications correctes avec des données brutes, nous avons jugé pertinent d'essayer également avec les données brutes sans pré-traitement.

Ainsi, cette méthode est implémentée des deux façons respectivement dans les dossiers "alexnet_stft" et "alexnet_raw".

Les données d'entrée d'une architecture alexnet sont sous forme d'images 2D avec un nombre défini de canaux.

Le spectrogramme d'un son brut est une image 2D représentant les fréquences d'un son dans le temps. L'intensité d'un pixel correspont à l'amplitude d'une fréquence, les abscisses sont la valeur de cette fréquence et les ordonnées représentent l'instant où cette fréquence est présente dans le son.

Pour les données brutes, le son sera simplement découpé en trames pour former une image 2D dont la largeur fait la taille de la trame.

Cette méthode étant inspirée du transfert de style d'images, elle requiert des données d'entrées avec des canaux. Dans AudioStyle Transfer[7], il est suggéré de dupliquer les données d'entrée sur les différents canaux pour simuler les canaux RGB. Le nombre de canaux peut être donc choisi arbitrairement.

3.2.4 Entrainement

Les classifieurs sont entrainés sur l'ensemble "nsynth-train". Par manque de mémoire vive et de puissance calcul, le classifieur de données brutes a été entrainé avec 4000 échantillons, tandis que le classifieur de spectrogrammes est entrainé sur 1000 échantillons. Cependant, l'entrainement du classifieur de spectrogramme étant trop lent pour pouvoir l'inclure dans les délais du stage, son architecture a été réduite à la première couche de convolution pour accélérer l'entrainement.

3.2.5 Extraction du style

Tout d'abord, les activations de la dernière couche de convolution du réseau sont extraites après passage d'un son dans le réseau. L'opération est schématisée sur la figure 3.4.

Ces activations sont extraites sous forme d'un tableau en 3 dimensions, comparable encore une fois à une image 2D avec des canaux.

La différence de style est calculée en plusieurs étapes. D'abord, on construit des matrices de Gram pour chaque combinaison possible de channels k :

$G_{ij} = \sum_k A_{ik} A_{jk}$.

On stocke ensuite ces matrices de Gram dans une plus grande matrice qui fait la hauteur et la largeur de l'image 2D : la matrice de style.

3.2.6 Transfert de style

Le transfert de style va se faire d'un son à l'autre par descente de gradient. L'objectif sera de minimiser la différence entre deux matrices de style.

Contrairement à la méthode de Gatys [1] et conformément aux conclusions de "Audio Style Transfer" [7], nous n'utiliserons pas le "contenu" et nous initialiserons la descente de gradient avec le son B.

La valeur à minimiser est la suivante :

$Styleloss = \frac{1}{4N^2M^2} \sum_{ij} (M^A_{ij} - A^B_{ij})$

Avec M^A et M^B matrices de style des sons A et B respectivement.

La fonction à minimiser prend donc en entrée le son B et le fait passer dans le réseau de neurones pour extraire sa matrice de style. La matrice de style du son A est précalculée et sert à calculer la perte de style entre les deux sons.

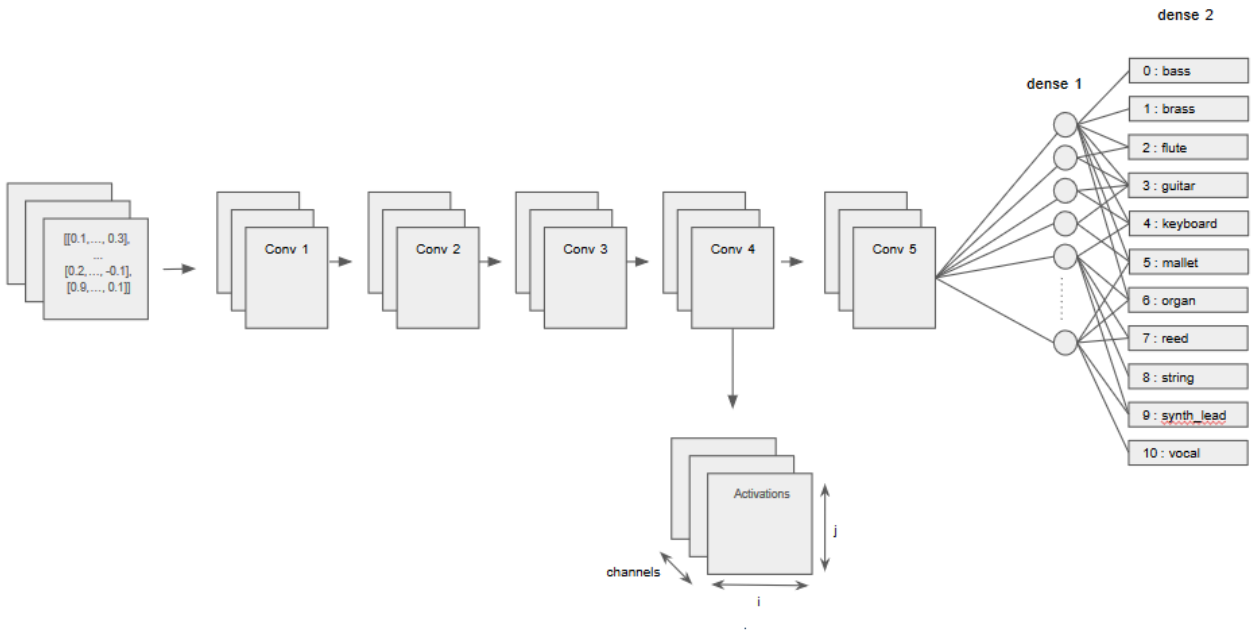

FIGURE 3.4 – Schéma de l'extraction d'une matrice de style du réseau de neurones. La méthode consiste à découper le réseau et à se servir de la convolution 4 comme sortie à la place de la couche dense 2.

3.2.7 Résultats

3.2.7.1 Qualité des classifieurs

Le classifieur utilisant les données brutes classe 62,2% des sons correctement sur 1000 données de validation évaluées.

Le classifieur de spectrogrammes donne une classification moins satisfaisante de 11,2% sur 1000 données de validation évaluées. On peut considérer qu'il n'est pas suffisamment entrainé pour être utilisé dans le transfert de style.

3.2.7.2 Transfert de style

La figure 3.5 présente les résultats de transfert.

La bibliothèque "optimize" de "scipy" permet d'effectuer facilement des descentes de gradient avec des tableaux en entrée.

Après plusieurs heures d'attente, la fonction ne semble pas modifier suffisamment le son d'entrée et la valeur de style ne diminue pas. Le son sortant est quasiment identique au son d'entrée.

De même, une tentative avec la bibliothèque "pyswarm", utilisant un systeme de "photons swarm" pour minimiser des tableaux, s'avère infructueuse et génère beaucoup de bruit sur le son.

Ces résultats ne sont pas très satisfaisants [1]. Cependant, l'article original de transfert sur les images indique qu'ils utilisent les gradients du réseau de neurones pour accélérer la minimisation. Je n'ai pas réussi à l'implémenter sur ma version de l'algorithme, mais cette méthode sera utilisée avec l'implémentation du VGG19.

3.2.7.3 Etude du style

Intuitivement, la méthode proposée prétend que la matrice de style calculée précédemment, étant extraite d'un réseau entrainé à classifier des instruments, correspond à une information abstraite pertinente au choix de la classe à assigner. Ainsi, elle devrait être similaire pour deux instruments de la même classe, puisqu'elle génère le même résultat dans les couches suivantes du réseau. La différence entre les matrices de deux instruments

1. Sons disponibles sur http ://nicho.fr/transferotron

FIGURE 3.5 – Spectrogrammes de gauche à droite : son de guitare, son de clarinette, résultat du transfert par descente de gradient, résultat du transfert par nuée de photons

identiques devrait être proche de zéro. De même, des instruments différents devraient avoir des matrices de style éloignées et donc une différence élevée.

Pour vérifier ces hypothèses, deux tests ont été mis en place :

— Variation de la note d'un instrument

— Variation de l'instrument sur une même note

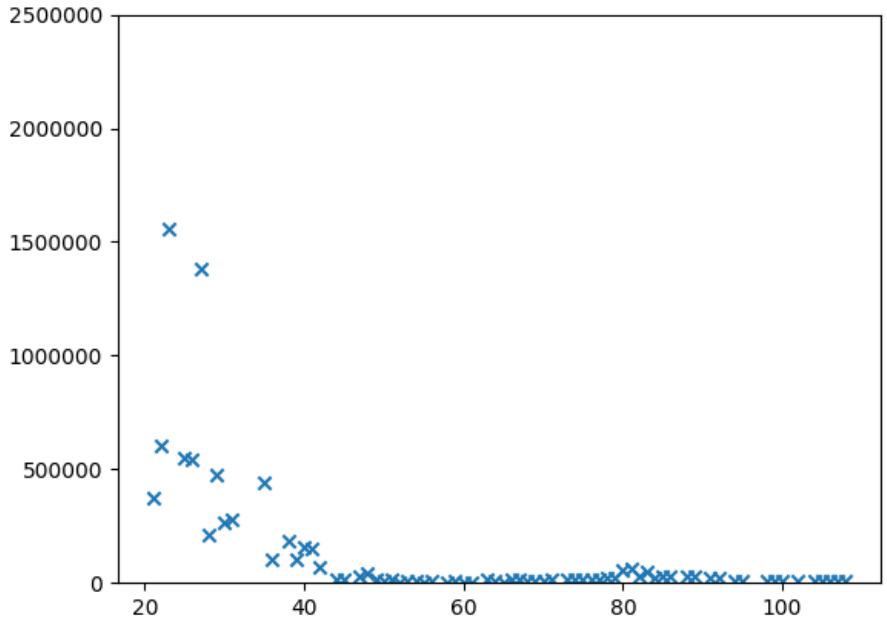

FIGURE 3.6 – Différence de style entre le fichier "guitar_acoustic_014-058-075.wav" et le même instrument jouant des notes différentes (indiquées en abscisse)]

La figure 3.6 représente la variation de notes en conservant le même instrument. On voit que le style est éloigné de 0 pour les notes précédant la note 42, mais qu'il est très proche de 0 pour toutes les notes de la partie

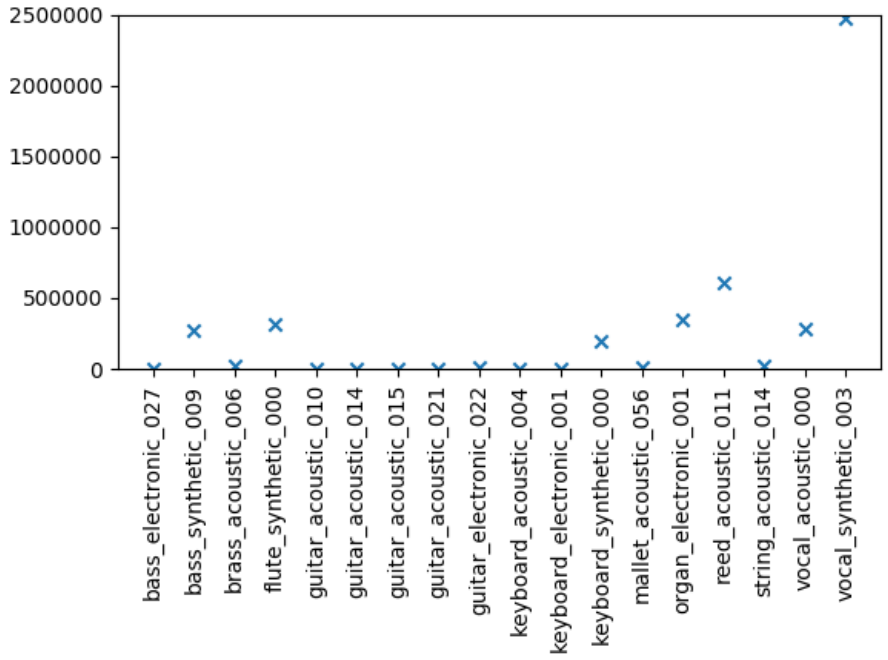

FIGURE 3.7 – Différence de style entre le fichier "guitar_acoustic_014-058-075.wav" et d'autres instruments jouant la note 057.

droite. La partie de droite semble respecter l'idée que la différence de style est faible, voire nulle, entre deux notes d'un même instrument. On pourrait expliquer la partie de gauche en écoutant les sons qui la composent. Les plus graves semblent être beaucoup plus saturés et de moins bonne qualité que les plus aigues : cela pourrait influer sur le résultat.

La figure 3.7 représente une variation d'instrument sur une note identique. On remarque que des instruments appartenant à la même famille, ou au même type d'instrument (guitares, basses), affichent une différence de style proche de 0. Il y a toutefois quelques accrocs : deux claviers, une corde et le cuivre affichent aussi une différence de style proche de 0. Cependant, la plupart des autres instruments affichent bien une différence non nulle, et aucune guitare n'est séparée du groupe.

Ces quelques résultats mériteraient d'être étendus sur l'ensemble de la base de donnée, en utilisant les descripteurs fournis par NSynth pour analyser les données obtenues. Cependant cette opération est trop couteuse en temps. Ces quelques résultats semblent concorder avec l'intuition que la matrice de style correspond bien à une information abstraite liée à la classe de l'instrument.

3.3 VGG19

Les version précédentes de l'algortihme ne parvenant pas à converger vers des résultats, j'ai repris l'exemple de code de transfert de style proposé dans les exemples du github de Keras [2].

3.3.1 Méthode

Le principe de l'algorithme est le même que précédemment, mais il introduit quelques nouveautés (figure 3.8).

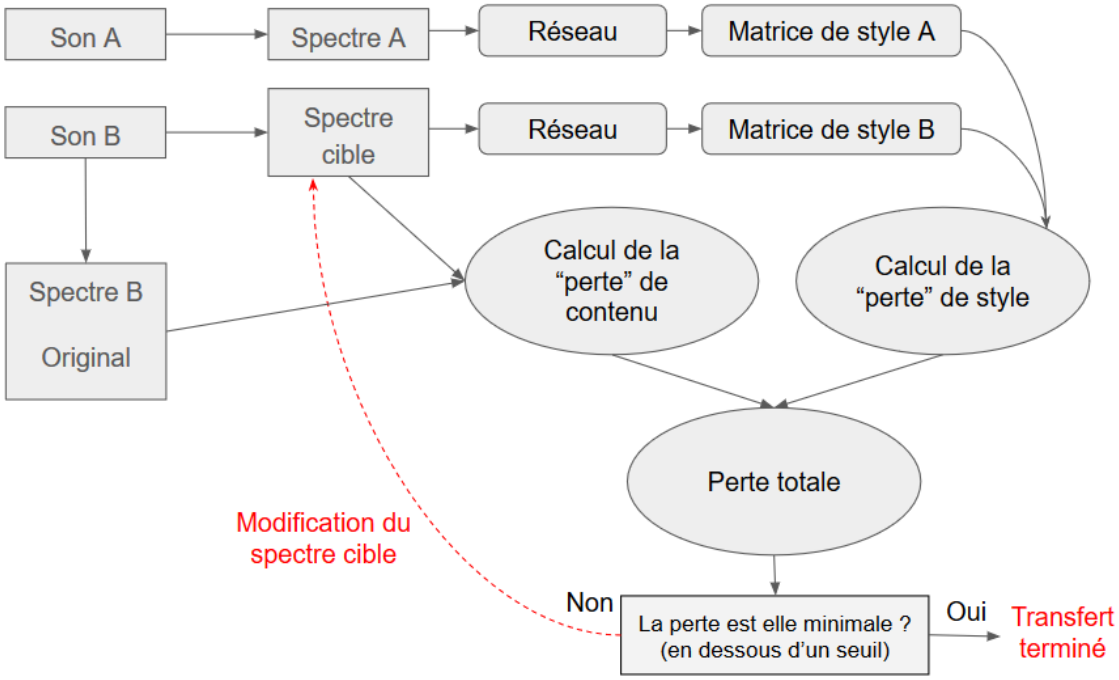

FIGURE 3.8 – Nouvelle version de la méthode de transfert de style

La nouvelle fonction à minimiser contient toujours la différence de style entre la cible et le son dont on transfère le style, mais aussi la différence de contenu entre la cible modifiée et son originale. Les données sont désormais des spectrogrammes d'amplitude.

3.3.2 Architecture

On utilise l'architecture VGG19 3.9 qui est originellement destinée à la classifications d'image. Entrainée sur 1000 classes d'images, c'est cette architecture qui est utilisée dans l'algorithme de transfert de style d'images de Gatys[4].

3.3.3 Entrée

Les entrées, comme pour la deuxième version de l'AlexNet, sont des spectrogrammes d'amplitude des sons. On les duplique sur les 3 canaux RGB pour correspondre à l'architecture.

Le réseau VGG19 nécessite cependant que les entrées correspondent à son format d'entrée. Il propose pour celui une fonctione de pré-processing mais la documentation n'est pas claire sur ce qu'elle effectue. Il faudra passer par une étape de post-processing pour annuler les effets de ce formatage.

2. https ://github.com/keras-team/keras/blob/master/examples/neural_style_transfer.py

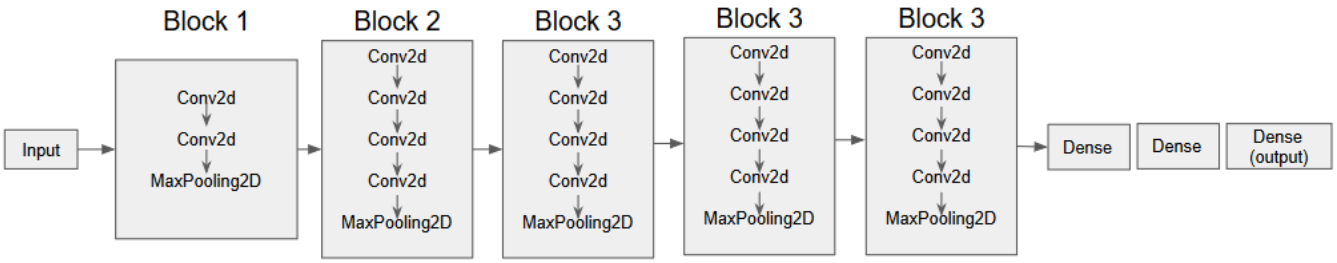

FIGURE 3.9 – Architecture vgg19

3.3.4 Entrainement

Le réseau est pré-entrainé à reconnaitre des images. D'après "Audio style transfer"[7], les résultats d'une telle architecture sont satisfaisants bien qu'elle ne soit pas entrainée à reconnaitre des sons.

Ce même article prétend qu'un réseau dont les poids sont initialisés aléatoirement permet des transferts de style satisfaisants.

Pour tester ces hypothèses, trois versions de ce réseau ont été mises en place :

— Réseau pré-entrainé avec des images
— Réseau Initialisé aléatoirement
— Réseau entrainé avec des spectrogrammes

Une version légèrement modifiée du code source de VGG19 permet de le réentrainer avec de nouvelles données. L'entrainement, de la même façon que pour ma première version avec AlexNet, ne semble pas augmenter l'efficacité de la classification au cours des époques. En observant les différents spectrogrammes sous forme d'images, on peut voir qu'ils sont très uniformes et les différences ne sont peut être pas suffisamment fortes pour les discerner. J'ai essayé de convertir les spectrogrammes en RGB pour voir si cela permettait d'améliorer l'entrainement mais cela n'a pas abouti.

L'augmentation de la taille de l'ensemble d'entrainement a eu pour effet d'améliorer légèrement la précision du réseau. Il est probable que ce soit la solution à ce problème. Il s'agit ici d'une limite car le serveur du CREMI manque de mémoire vive à partir de 2000 échantillons seulement. C'est très loin des 280 000 échantillons d'entrainement que pourrait fournir NSynth.

De plus, le réseau met déjà plusieurs jours à s'entrainer avec 2000 échantillons, ce qui n'irait qu'en augmentant avec un ensemble plus gros.

3.3.5 Transfert de style

La différence de contenu est calculée par une distance au carré entre la cible modifiée et son originale.
$Loss_{Contenu} = \frac{1}{2}\sum_{i,j}(Oij - Cij)^2$

Avec O et C respectivement le spectrogramme du son orginial et le spectrogramme modifié.

On fournit ensuite à la fonction de descente de gradient les gradients du réseau pour lui permettre de converger plus rapidement, dans un mécanisme similaire à la back-propagation.

Les spectrogrammes sont inversibles et permettent donc de reconstruire le son après modification du spectre, à la fin de la descente de gradient.

3.3.6 Visualisation

La génération et l'observation de sons multiples en parallèle se sont révélées assez fastidieuses. Pour palier à ce problème, le script "generate.py" et ses variantes permettent de générer automatiquement toutes les combinaisons de transfert de style entre tous les fichiers ".wav" d'un dossier.

Pour permettre une navigation plus simple de ces sons générés, le programme "gui.py" (figure 3.10) affiche automatiquement les spectrogrammes des fichiers sources ainsi que le spectrogramme du fichier généré. Il permet également d'écouter les sons.

Le panneau de gauche permet de sélectionner un dossier, de configurer le type de génération à lire (pré-entrainé, initialisé aléatoirement, spectrogramme de phase au lieu d'amplitude...) ainsi que de sélectionner les

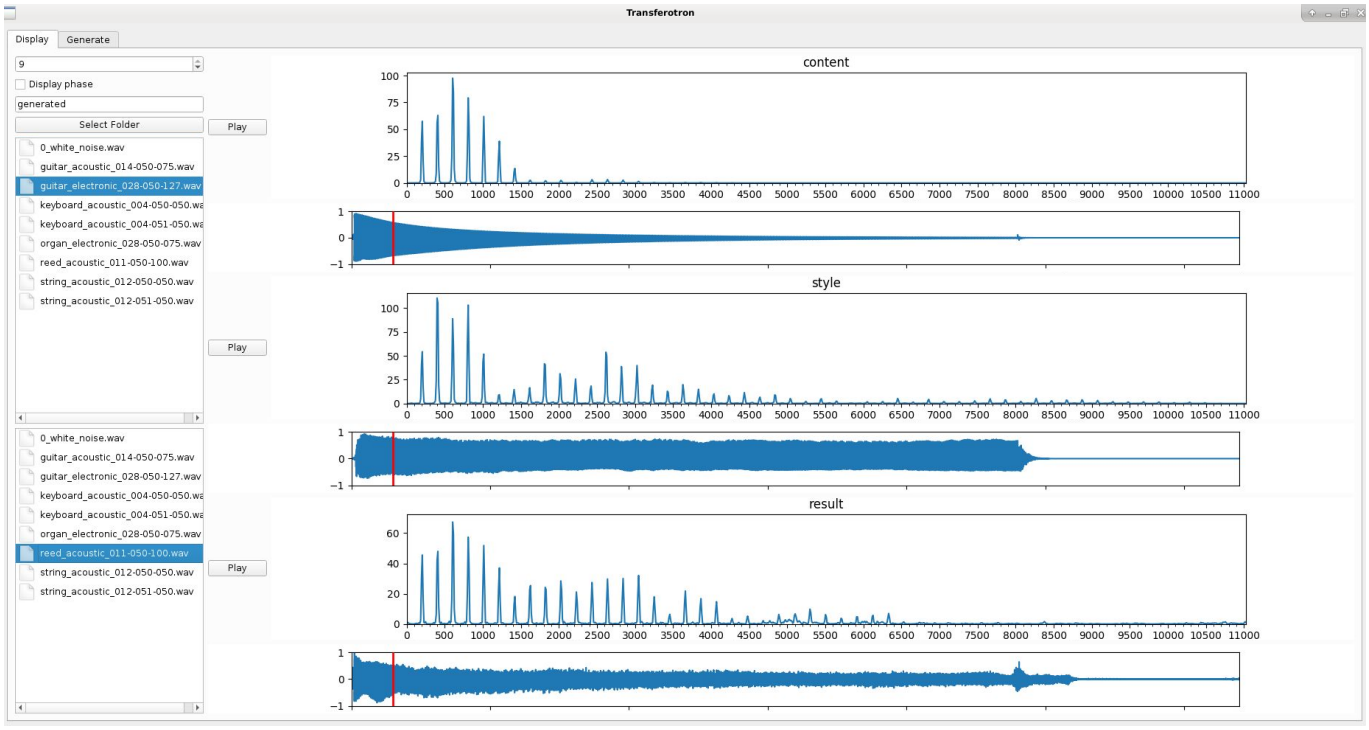

FIGURE 3.10 – Interface du visualisateur de spectrogrammes.

deux fichiers à combiner. Biensur, les sons résultats doivent être précalculés avec le programme "generate.py" pour apparaitre, sinon un graphe vide s'affiche.

Le graphe affiché est un spectrogramme sur une des trames temporelles de la transformée de fourier. Ce sont donc les fréquences présentes dans le son à un instant donné.

La forme brute du son est affichée en dessous, avec un curseur permettant de se déplacer dans le temps.

3.4 Résultats

3.4.1 Version pré-entraînée sur des images

Les résultats de cette version sont les seuls à être vraiment audibles. L'objectif de ces tests sera à la fois d'essayer de générer des sons subjectivement intéressants, mais aussi de tester des cas particuliers pour voir s'ils nous permettent de mieux comprendre ce que fait le transfert. Il est conseillé d'écouter les sons en plus de regarder les spectrogrammes [3]. De même, la plupart des sons et des spectrogrammes ne sont pas présents ici pour éviter de ne faire qu'une énumération.

3.4.1.1 Identité

Tout d'abord, si l'algorithme fonctionne correctement, appliquer le style d'un son sur le contenu du même son devrait donner un son identique.

Les résultats de la figure 3.11 et l'écoute des sons montrent que l'identité reproduit bien le son d'origine, mais elle le bruite en exacerbant les pics déjà présents dans le spectrogramme. Cela est probablement dû au fait que malgré qu'elle ait rencontré une perte de style de 0, la descente de gradient continue à faire ses itérations. Elle pourrait donc essayer aléatoirement de diminuer la perte en vain et créer ces bruits.

3.4.1.2 Résultats intéressants à l'écoute

Ces deux résultats (figures 3.12 et 3.13) sont plutôt agréables à l'oreille (relativement aux autres sons générés) et donnent vraiment l'impression de combiner des caractéristiques des deux instruments à la fois.

3. Sons disponibles sur http ://nicho.fr/transferotron

FIGURE 3.11 – De gauche à droite : son de guitare, identité entre deux sons de guitare, son de piano, identité entre deux sons de piano

FIGURE 3.12 – De gauche à droite : son de guitare, son de clarinette, contenu de la guitare avec le style de la clarinette, contenu de la clarinette avec le style de la guitare

Sur la figure 3.12, il semblerait que le style définit plutôt l'enveloppe spectrale des pics générés tandis que le contenu influence l'intensité des pics. C'est aussi le cas le sur 4e spectrogramme de la figure 3.13. Le 3e spectrogramme ne semble pas avoir l'enveloppe du second, mais sa fréquence fondamentale est la même que sur le 1er.

Le point commun de ces deux exemples est qu'ils sont une combinaison d'un son net avec une attaque puissante et d'un son tenu à attaque faible. Le fait qu'ils soient un peu plus audibles que la plupart des autres est peut-être une piste pour mieux utiliser le transfert de style.

3.4.1.3 Bruit blanc

La méthode de transfert de style d'images utilise le bruit blanc pour générer des "textures" représentant le style d'une image. Ici, on essaie de faire la même chose (figure 3.14).

Lorsqu'on transfère le style de la guitare sur du bruit blanc, on obtient une image un peu bruitée avec des pics apparents à des endroits aléatoires. Ce n'est pas étonnant : le réseau étant entrainé à reconnaitre des images, le comportement de transfert sur du bruit blanc est le même qu'en image. On a donc appliqué la "texture"

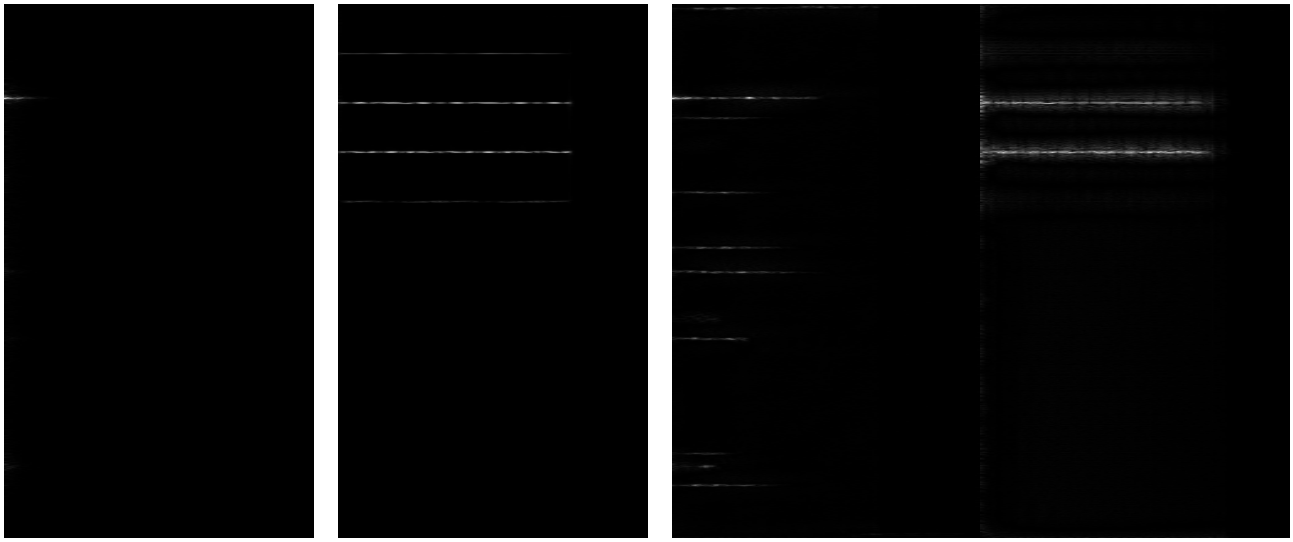

FIGURE 3.13 – De gauche à droite : son de xylophone, son d'orgue, contenu de xylophone avec style d'orgue, contenu d'orgue avec style de xylophone

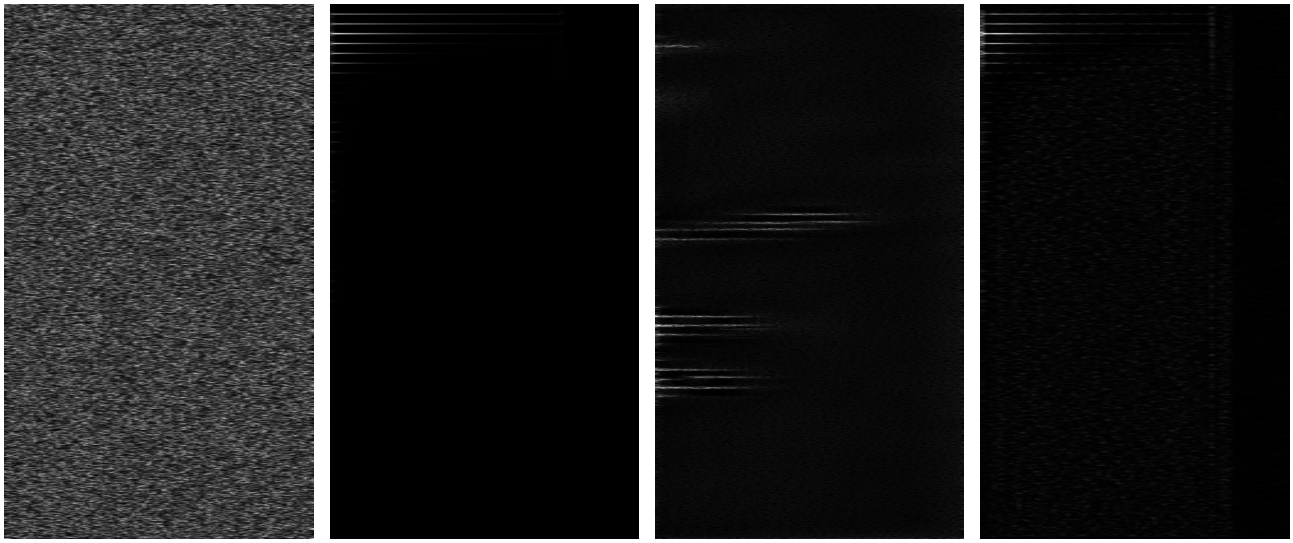

FIGURE 3.14 – De gauche à droite : bruit blanc, guitare électrique, contenu de bruit blanc avec style de guitare électrique, contenu de guitare électrique avec style de bruit blanc

visuelle (les pics) de façon aléatoire sur l'image.

De même, le transfert du style de bruit blanc sur la guitare n'a que pour effet de bruiter le son.

3.4.1.4 Silence

De la même façon que pour le bruit blanc, il est intéressant de voir comment va se comporter la méthode avec du silence (figure 3.15). Ici, on peut observer plus facilement les résultats en les écoutant.

Il semblerait qu'appliquer le style du silence sur un son lui donne un aspect "étouffé". Appliquer un style quelconque sur un silence semble générer une copie des pics en bas de l'image, donc dans des fréquences plus élevées.

3.4.1.5 Pics sinusoïdaux

Ici, l'objectif était de voir si rajouter une fréquence dans un son allait avoir une influence sur le transfert. Cependant, ces résultats ne sont pas exploitables car les sons de guitare modifiés ont introduit de la saturation

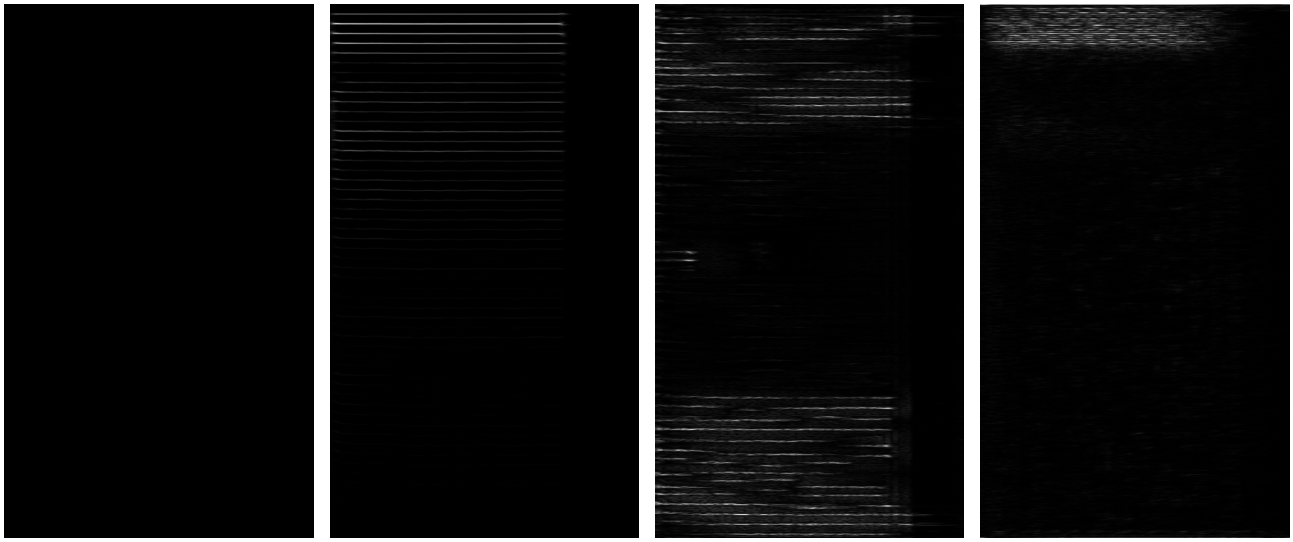

FIGURE 3.15 – De gauche à droite : silence, clarinette, contenu de silence sur style de clarinette, contenu de clarinette sur style de silence

et des bruits non désirés en plus des sinusoïdes. Les sons sont tout de même disponibles à l'écoute.

3.4.1.6 Prétraitement d'intensité

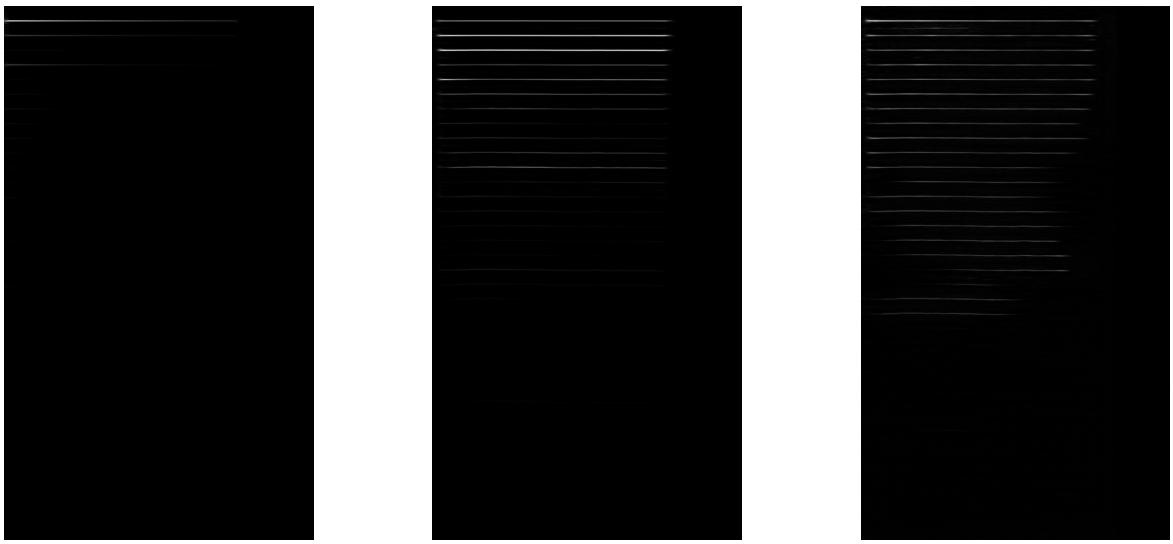

FIGURE 3.16 – De gauche à droite : guitare, clarinette, contenu de guitare avec style de clarinette

Afin de rendre les sons plus agréables à l'écoute, et d'effacer les artéfacts qui étaient générés dans les espaces de silence, un prétraitement qui consistait à appliquer au son de sortie les mêmes intensités que le son de style a été mis en place. Le résultat est clairement plus audible mais on perd beaucoup de caractéristiques du son de contenu.

3.4.1.7 Etude du style

Parallèlement à l'étude du style pour l'architecture AlexNet, l'objectif ici est de vérifier si la différence de style calculée correspond bien à des différences d'instruments et donc si des instruments identiques génèrent bien une différence faible.

La figure 3.17 semble indiquer qu'il y a bien un lien entre la différence de style et la note jouée par l'instrument. En revanche, sur la figure 3.18, on voit que le changement d'instrument n'a pas beaucoup d'effet sur la

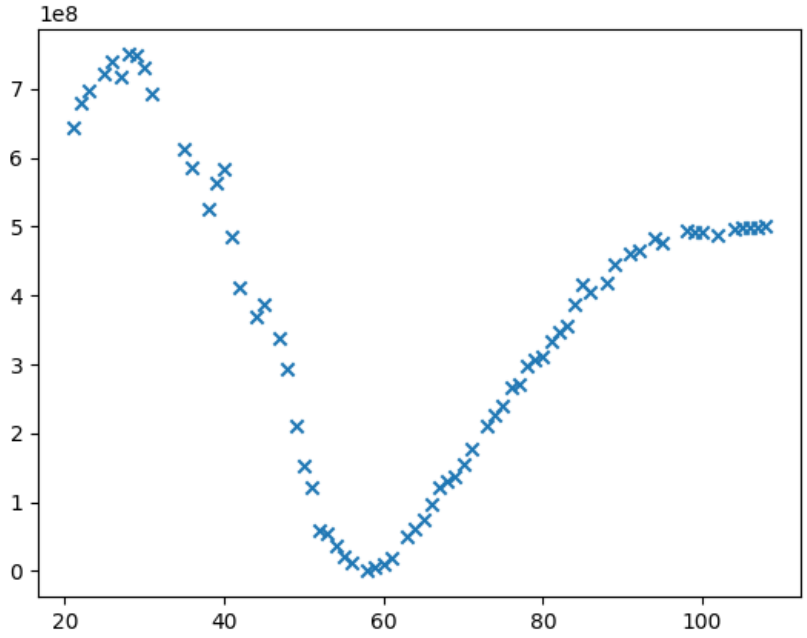

FIGURE 3.17 – Différence de style entre le fichier "guitar_acoustic_014-058-075.wav" et le même instrument jouant des notes différentes (indiquées en abscisse)

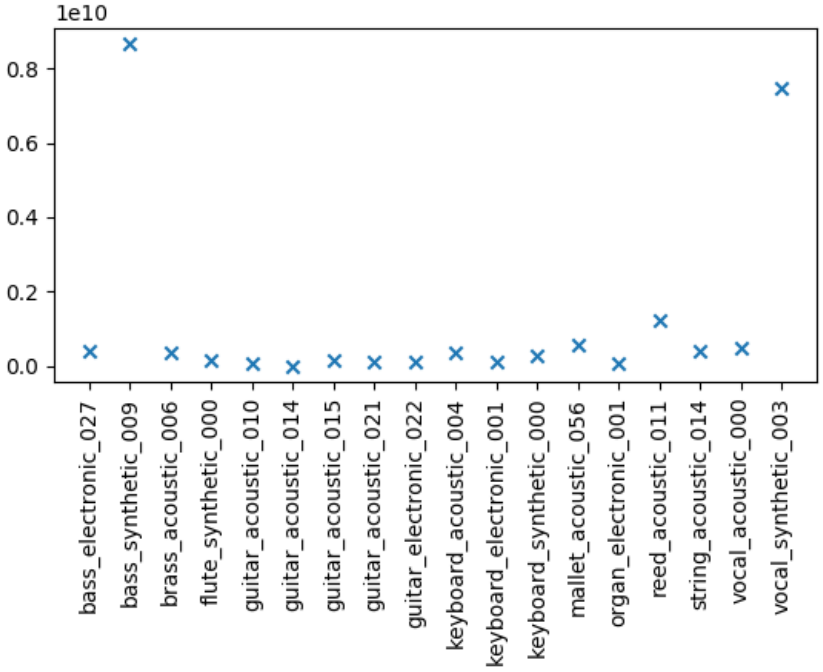

FIGURE 3.18 – Différence de style entre le fichier "guitar_acoustic_014-058-075.wav" et d'autres instruments jouant la note 057.

différence de style (à l'exception des voix).

Il semblerait donc que contrairement au réseau entrainé sur des sons, celui-ci ne permet pas de fournir une information abstraite qui nous permettrait de différencier des instruments entre eux. Cependant, cette différence de style n'est pas dénuée d'information : elle contient probablement des informations relatives à la note jouée.

Chapitre 4

Bilan

La synthèse de sons par réseau de neurones est une discipline qui prend beaucoup de temps et de ressources. La plupart des applications actuelles sont des applications de "text-to-speech", mais le milieu de la synthèse musicale se développe petit à petit. Au cours de ce stage, j'ai pu me familiariser avec l'entrainement et l'utilisation de réseaux de neurones profonds. L'avantage de commencer avec un problème comme celui du transfert de style, c'est qu'il m'a poussé à essayer de comprendre ce qui se passe à l'intérieur du réseau au lieu de l'utiliser comme une boite noire "magique".

Les résultats que j'ai obtenus sont parfois satisfaisants à écouter, mais de manière générale j'aurai préféré obtenir plus de résultats avec des versions entrainées du réseau, des résultats que je serai peut-être parvenu à expliquer, à évaluer.

Faute de temps et de puissance de calcul, la plupart de mes expériences n'ont pas été effectuées sur assez d'échantillons pour en déduire des conclusions solides. Cependant, dans l'ensemble, ce qu'on peut en retenir, c'est que le transfert de style est une methode qui peut donner des résultats très différents en fonction des choix que l'on fait. En utilisant VGG19, on obtient des résultats satisfaisants à l'oreille mais il est difficile d'expliquer quelle caractéristique du son a été transférée. La structure AlexNet avec les enveloppes temporelles produit une bonne classification et l'étude du style semble indiquer que l'information transférée est cohérente avec l'entrainement du réseau. Cependant, sans mécanisme de conservation des phases lors de la reconstruction de l'enveloppe temporelle, il est très probable que même si la descente de gradient converge, le résultat final soit plein de clics et de déphasages.

Chapitre 5

Perspectives

Lors de la présentation de cette méthode et des sons générés à un public d'informaticiens et de musiciens, la question qui revenait le plus souvent était "Qu'est-ce que le style d'un son ?". En effet, même si la méthode VVG19 pré-entrainée génère des sons satisfaisants à l'écoute, au cours de laquelle on peut subjectivement se dire qu'il s'agit bien d'un mélange des deux sons, il est difficile de dire précisément quelle partie du son a été transférée, même en observant les spectrogrammes.

La plupart des articles proposant des méthodes de transfert de style présentent des résultats qu'ils considèrent satisfaisants à l'écoute, ou observent des similarités entre les spectrogrammes des sons initiaux et du son généré.

Pour évaluer efficacement ce genre de résultats, il est nécessaire de définir exactement ce qu'est le "Style" et de concevoir des outils permettant de le vérifier.

Un idée serait d'abandonner complètement le concept de "Style" au profit d'une simple "Caractéristique" que l'on transfèrerait. La méthode est similaire : on entraine un réseau à reconnaitre cette caractéristique puis on applique le même algorithme de transfert que pour la version AlexNet, avec seulement la différence de "Style".

Le résultat est un nouveau son, sur lequel on a transféré la caractéristique que le réseau sait reconnaitre. Un passage de ce nouveau son dans le réseau entrainé permettrait, s'il se retrouve classé correctement, de confirmer le transfert de la caractéristique.

Ainsi, la définition de la caractéristique transférée n'est plus établie par convention mais par rapport au réseau entrainé. Il s'agit donc d'entrainer le plus précisément possible un réseau à reconnaitre ce qu'on voudrait transférer.

De même, en combinant plusieurs réseaux reconnaissant des caractéristiques différentes, il serait éventuellement possible, en ajoutant leurs "différences de caractéristiques" à la différence à minimiser, de faire converger une descente de gradient pour conserver plusieurs caractéristiques à la fois. Au final, la présence de ces caractéristiques dans le son généré peut être confirmée par un classement dans les réseaux correspondants.

Un éventuelle application serait de pouvoir transférer le timbre d'un instrument sur un autre, en conservant la note. Cela pourrait permettre par exemple de récupérer le timbre d'un joueur de guitare directement dans un morceau pour l'appliquer sur ses propres enregistrements et profiter des mêmes effets sonores.

Les pistes d'amélioration à ces algorithmes sont nombreuses. Utiliser le réseau WaveNet, changer de type de données entrantes (cepstre ?), entrainer le réseau plus longtemps sur des ensembles de données plus grands.

L'approche temporelle du AlexNet pourrait elle aussi être améliorée en étudiant un peu plus l'algorithme de back-propagation et en s'en servant pour optimiser la descente de gradient.

Bibliographie

[1] Shuqi Dai, Zheng Zhang, and Gus Xia. Music style transfer issues : A position paper. *CoRR*, abs/1803.06841, 2018.

[2] Chris Donahue, Julian McAuley, and Miller Puckette. Synthesizing audio with generative adversarial networks. *CoRR*, abs/1802.04208, 2018.

[3] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. *CoRR*, abs/1704.01279, 2017.

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[7] Eric Grinstein, Ngoc Duong, Alexey Ozerov, and Patrick Perez. Audio style transfer. 10 2017.

[8] Parag K. Mital. Time domain neural audio style transfer. *CoRR*, abs/1711.11160, 2017.

[9] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. *CoRR*, abs/1805.07848, 2018.

[10] Se Rim Park and Jinwon Lee. A fully convolutional neural network for speech enhancement. *CoRR*, abs/1609.07132, 2016.

[11] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet : A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

[12] Prateek Verma and Julius O. Smith. Neural style transfer for audio spectograms. *CoRR*, abs/1801.01589, 2018.