

Composition Musicale par Réseau de Neurones

AUFFRET Guillaume
GOSSELIN Pierre
LOUZI Idriss

2023

Table des matières

1	Introduction	3
2	Réseau de neurones	4
2.1	Une approche mathématique	5
2.1.1	L'apprentissage	5
2.2	L'architecture LSTM	6
2.3	Implémentation en Python avec Keras	7
3	Format MIDI	9
3.1	Présentation	9
3.2	Structure du fichier	9
3.3	Implémentation en Python	11
3.3.1	représentation en octets	11
3.3.2	Fonctions de passage du format MIDI au jeu de données	12
4	Jeu de données Maestro	13
4.1	Présentation du jeu de données	13
4.2	Analyse statistique du dataset [Num06]	13
4.2.1	Valeur des notes	13
4.2.2	Durée des notes	15
4.2.3	Durée avant la prochaine note	15
4.3	Prévision	16
4.3.1	Prévision à une note	16
4.3.2	Prévision à deux notes [Col]	17
4.3.3	Prévision à plus de deux notes	17
4.4	Influence des compositeurs	18
5	Jeux de données extraits du jeu de données Maestro	20
5.1	Présentation des jeux de données	20
5.2	Jeu de données MINI1-dataset	20
5.3	Jeu de données MINI10-dataset	24
5.4	Jeu de données complet dataset-16notes	28
5.5	Conclusions à tirer	32

6	Entraînement du réseau de neurones	33
6.1	Les données	33
6.2	Choix du modèle	34
6.2.1	Analyse des courbes d'apprentissage (Annexe 1)	36
6.3	Amélioration	38
7	Génération de musique	40
7.1	Données en jeu	40
7.2	Analyse	40
7.2.1	Valeur des notes	40
7.3	Durée des notes	42
7.4	Durée entre les notes	44
7.5	Conclusion sur les résultats	45
8	Annexe 1 : Courbes d'apprentissage	47

1 Introduction

L'intelligence artificielle s'ouvre à de nouvelles perspectives à mesure de son essor dans le monde d'aujourd'hui, notamment dans le domaine de la création artistique. Notre projet s'inscrit pleinement dans ce secteur ci : nous avons pour objectif de créer de la musique en utilisant un réseau de neurones récursif.

Les enjeux de ce projet sont multiples et d'importance capitale pour notre parcours académique et professionnel : En effet, dans un premier temps ce projet nous permettrait d'acquérir des compétences poussée en python (un langage omniprésent dans les débouchés de la filière génie mathématiques) en explorant ses fonctionnalités avancées pour le développement d'application liée à l'intelligence artificielle.

De plus il s'agit d'un premier contact avec les technologies et concepts clés de l'intelligence artificielle. Etant donné l'importance de l'IA de nos jours, ce projet nous facilitera la compréhension de notions plus complexes que nous pourrions être amenés à employer dans un cadre professionnel.

Nous nous sommes orienté vers ce projet plutôt qu'un autre pour différentes raisons : D'abord, il y a le fait que ce projet diffère radicalement de ce que nous déjà pu découvrir en GM3. Ensuite, la mise en relation de l'intelligence artificielle et de la musique relève de l'intérêt personnel, ce sont des notions pour lesquelles nous avons un intérêt commun. Et enfin, nous avons tous les trois l'ambition de travailler avec les technologies de l'intelligence artificielle, c'est donc l'occasion pour nous de les découvrir. D'autres se sont déjà prêté à l'exercice [Joh15] [Ten21] et nous proposerons notre propre implémentation.

Notre objectif final sera de produire des résultats concrets (compréhensible et comparable à la base de données choisie) que nous pourrons analyser. Nous pourrons alors discuter de la pertinence et de la qualité du processus de création musicale automatisée.

2 Réseau de neurones

Dans cette section, nous définirons l'architecture d'un réseau de neurones que nous implémenterons et exploiterons par la suite dans le cadre de ce projet.

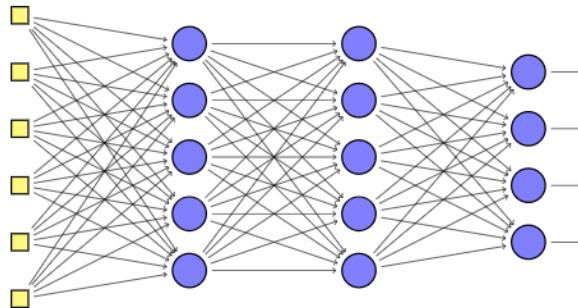


FIGURE 1 – Réseau de neurones feedforward

Un réseau de neurones est un modèle mathématique et informatique inspiré directement du cerveau humain [Liu23]. Il se décompose en plusieurs couches, elles-mêmes composées de plusieurs unités de calcul nommées "neurones virtuels". Un réseau de neurones est composé de différents types de neurones.

Tout d'abord, les neurones de la première couche reçoivent des données extérieures "brutes". Chacun des neurones traite indépendamment une partie de l'information. En fonction de l'information reçue, un neurone s'active ou non (nous reviendrons sur les conditions d'activations). Lorsqu'un neurone s'active, il effectue alors un traitement consistant à l'ajout d'un biais propre à ce neurone, il transmet ensuite la nouvelle information aux neurones de la couche suivante.

Les neurones des couches intermédiaires ("hidden layers") reçoivent les informations provenant des couches précédentes et effectuent à leur tour un traitement (s'ils sont activés), et ce processus se répète jusqu'à atteindre la couche finale.

Les neurones de la couche finale restituent alors une nouvelle information traitée.

Entre les différentes couches qui composent notre réseau se trouvent des connexions appelées synapses, elles lient les neurones entre eux. Ces dernières jouent un rôle dans l'activation d'un neurone puisque chacune d'entre elle est associée à un poids déterminant l'influence de la sortie d'un neurone sur l'autre. On dira que le poids module la transmission d'information.

Enfin le dernier traitement lors de la transmission d'une information d'un neurone à un autre : la fonction d'activation. Celle-ci normalise l'information traitée et détermine la valeur finale à transmettre à la couche suivante. On distingue plusieurs types de fonctions d'activation, parmi lesquelles les plus courantes sont :

- Neurones ReLu (Rectifier Linear Unit) : Ils s'activent proportionnellement à l'intensité des contributions activatrices, ils sont alors actifs lorsque l'information dépasse un certain seuil.

- Neurones sigmoïdes : Ils sont actifs uniquement si les contributions activatrices dépassent 50 pourcents (les rendants sensibles aux variations subtiles de l'information).

En résumé, le rôle d'un neurone est de collecter les contributions activatrices et inhibitrices provenant des neurones des couches précédentes et d'y ajouter un certain biais.

Notons que jusqu'ici nous avons décrit un réseau de neurone simple de type "feedforward", où l'information circule de manière unidirectionnelle. Dans le cadre de notre projet, nous exploiterons un réseau de neurone récursif (RNN), ce dernier est défini par une retransmission de l'information à la couche actuelle lors de la transmission à la couche suivante, permettant alors une meilleure capacité à traiter les données.

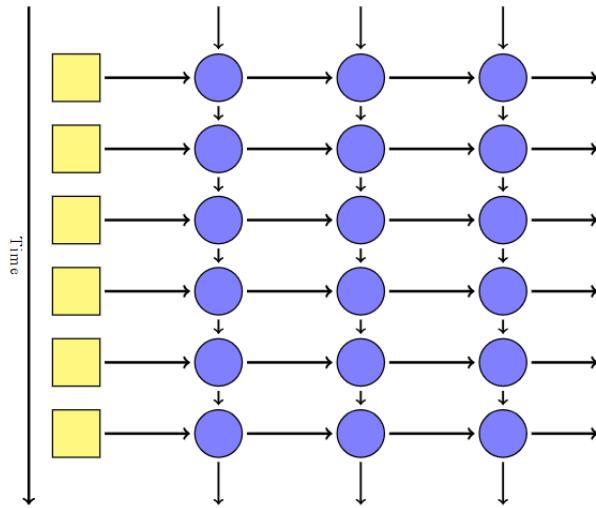


FIGURE 2 – Réseau de neurones récursif

2.1 Une approche mathématique

La théorie mathématique sous-jacente aux réseaux de neurones repose sur des concepts de l'algèbre linéaire, du calcul différentiel et de la théorie des fonctions. Les neurones virtuels dans un réseau de neurones sont essentiellement des unités de traitement mathématique. Chaque neurone effectue une combinaison linéaire des valeurs d'entrée, en utilisant les poids des connexions synaptiques, suivie d'une application d'une fonction d'activation non linéaire. Cette opération est essentielle pour introduire des non-linéarités dans le modèle, permettant ainsi au réseau de capturer des relations complexes dans les données.

On note w_i le poids porté par la synapse i , N le nombre de neurones portés par la couche précédentes (nombre de synapses entre les deux couches), x le vecteur d'information en entrée et B le biais associé à un neurone. La fonction d'activation est notée f . Le traitement effectué par un neurone se note alors $f(\sum_{i=1}^N w_i x_i + B)$.

Le résultat ainsi déterminé est à nouveau traité de la même façon par les neurones de la couches suivantes...

2.1.1 L'apprentissage

Une question reste en suspens : Comment déterminer efficacement les poids optimaux. Autrement dit comment apprendre à notre réseau à fournir un résultat cohérent présentant une erreur minimale ?

Présentons cela comme un problème d'optimisation, l'objectif étant alors de minimiser la fonction de perte donnée par : $\|X_{th} - X_{exp}\|_2^2$ avec X la sortie du réseau. Cette fonction (notons la $J(w_i)$) est appelée fonction de perte, l'idée est donc d'initialiser aléatoirement les poids w_i pour ensuite calculer le gradient de J nous indiquant alors dans quelle direction modifier les paramètres afin de le rendre le plus proche de 0 possible. A chaque itération, on modifie alors la valeur des paramètres jusqu'à finalement observer la convergence de notre méthode (Lorsque la perte est en dessous un certain seuil, l'algorithme s'arrête). Une fois l'apprentissage terminé, nous considérons nos paramètres optimaux et notre RNN capable de traiter les données de notre problèmes.

2.2 L'architecture LSTM

Il existe différentes architectures de Réseau de neurones récurrents présentant chacune différents avantages, l'architecture LSTM (Long Short Term Memory) a été conçue pour résoudre le problème de disparition du gradient.

Voici une présentation de l'architecture LSTM et de ses avantages :

Architecture LSTM :

1. Cellule LSTM : La principale composante d'un LSTM est la cellule LSTM elle-même. Elle est conçue pour maintenir et gérer l'information à long terme dans une séquence de données. Chaque cellule LSTM contient trois portes fondamentales :

- La porte d'oubli (forget gate) : Cette porte détermine quelles informations de l'état précédent de la cellule LSTM doivent être oubliées ou maintenues.
- La porte d'entrée (input gate) : Cette porte contrôle l'ajout de nouvelles informations à la cellule LSTM en fonction des données d'entrée.
- La porte de sortie (output gate) : Cette porte détermine quelles informations seront transmises en tant que sortie de la cellule LSTM.

2. État caché et état de cellule : Un LSTM maintient deux états internes, l'état caché (hidden state) et l'état de cellule (cell state). L'état caché contient les informations actuelles de la cellule, tandis que l'état de cellule stocke les informations à long terme.

3. Propagation dans le temps : Les cellules LSTM se propagent dans le temps, ce qui leur permet de conserver des informations de séquences antérieures tout en gérant efficacement les informations courantes.

Avantages de l'architecture LSTM :

1. Gestion des dépendances à long terme : L'un des principaux avantages des LSTM est leur capacité à gérer les dépendances à long terme dans les données séquentielles. Grâce à la porte d'oubli et à l'état de cellule, les LSTM peuvent stocker et accéder à des informations de séquences précédentes, ce qui les rend adaptés à des tâches telles que la prédiction de séquences temporelles.

2. Évitement du problème de la disparition du gradient : Les LSTM ont été spécifiquement conçus pour résoudre le problème de la disparition du gradient, qui affecte les RNN traditionnels. Grâce à leurs portes et à leur architecture, les LSTM peuvent apprendre des séquences à long terme sans que le gradient ne diminue de manière significative.

3. Flexibilité dans la modélisation : Les LSTM sont flexibles et peuvent être utilisés dans une variété de tâches, y compris la génération de texte, la classification de séquences, la traduction automatique et plus encore. Leur architecture modulaire permet également de les étendre pour répondre à des besoins spécifiques.

En résumé, l'architecture LSTM est une avancée significative dans le domaine des réseaux de neurones récurrents, car elle résout efficacement le problème de la disparition du gradient et permet de gérer des dépendances à long terme dans les données séquentielles. Ces avantages en font un choix populaire pour de nombreuses applications d'apprentissage automatique basées sur des séquences.

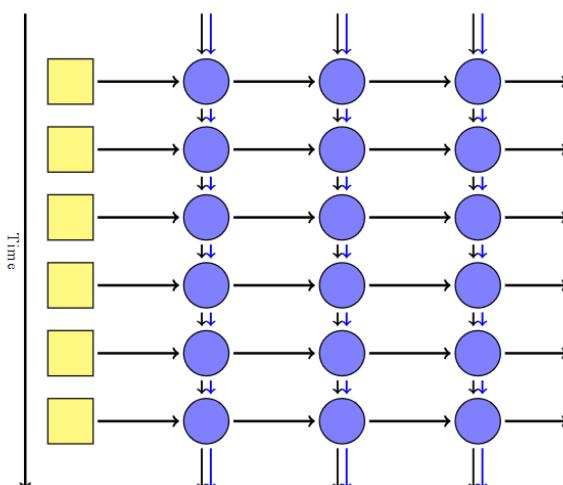


FIGURE 3 – Réseau de neurones récursif LSTM

2.3 Implémentation en Python avec Keras

Keras [Cho23] est une bibliothèque python, intégrée à TensorFlow [Ten15] (une autre bibliothèque python), permettant d'implémenter des modèles d'apprentissage.

Pour créer un tel modèle, deux options s'offrent à nous : la classe Model ou la classe Sequential. La classe Model permet de regrouper diverses couches d'apprentissage en un seul objet et permet ainsi de créer des modèles non linéaire complexes. Au contraire la classe Sequential met bout à bout différents modèles d'apprentissage de nature différente mais où la sortie du premier modèle sera l'entrée du second et ainsi de suite.

Nous utiliserons donc la classe Sequential, qui convient à notre problème.

L'architecture de notre séquence est la suivante :

1. - Une couche d'entrée qui permet seulement de renseigner la taille de la séquence de notes, et la dimension d'une note (le nombre de paramètres définissant une note)
2. - Une couche LSTM dont on spécifie la longueur, et elle prendra en entrée un vecteur de taille la dimension d'une note
3. - Une couche de sortie qui transforme la sortie du LSTM en une note (on a donc Dimension(note) neurones simples avec en entrée la sortie du LSTM)

On peut également ajouter des paramètres d'optimisation du modèle, afin d'éviter le sur-apprentissage :

1. Le nombre d'époques, c'est à dire le nombre de fois que le modèle va voir le jeu d'entraînement
2. Le nombre de Batch, c'est à dire le nombre de séquences que va rencontrer le modèle avant de corriger ses poids. Cela permet de paralléliser les calculs.
3. Batch Normalization : qui prends chaque séquence d'entrée et ajuste les paramètres (moyenne, écart-type...) par rapport à l'ensemble du jeu de données (il sera intéressant de savoir si cela baisse ou diminue la créativité artistique du modèle final)
4. Dropout : permet de mettre aléatoirement un certain pourcentage de poids en des poids nuls afin que le modèle retienne une tendance générale plutôt que d'apprendre par coeur le jeu de données
5. Early Stopping : arrête l'apprentissage du modèle lorsque l'écart des prédictions du modèle avec le jeu de test est au plus bas (on évite ainsi le sur-apprentissage du jeu d'entraînement)

Ainsi en python pour définir une séquence LSTM simple :

```

import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import InputLayer, LSTM, Dense, Dropout, BatchNormalization
from keras.callbacks import EarlyStopping

# _____
# Parametres du Dataset
# _____
nombreDeNotesPrecedentes = 7
formatNote = 3
# valeur de la note,
# dure, ,
# dure avant la prochaine note
# _____
# Creation du modele
# _____
nombreDeNeurones = 15          # nombre de neurones dans la couche LSTM
nombreDeBatch = 1               # nombre de subdivisions du jeu de donnees

# Creation d'un modele sequentiel
model = Sequential()

# Couche d'entrée
model.add( InputLayer(input_shape=(nombreDeNotesPrecedentes, formatNote)) )

# Couche LSTM
model.add( LSTM(nombreDeNeurones, activation='relu') )

# Couche de sortie
model.add( Dense(formatNote) )

# Compilation du modele avec
# la fonction de perte d'erreur quadratique moyenne
# l'optimiseur 'adam' qui est
# une descente de gradient stochastique
# a pas adaptatif
model.compile(loss='mean_squared_error', optimizer='adam')

# Afficher le resume du modele
model.summary()

```

3 Format MIDI

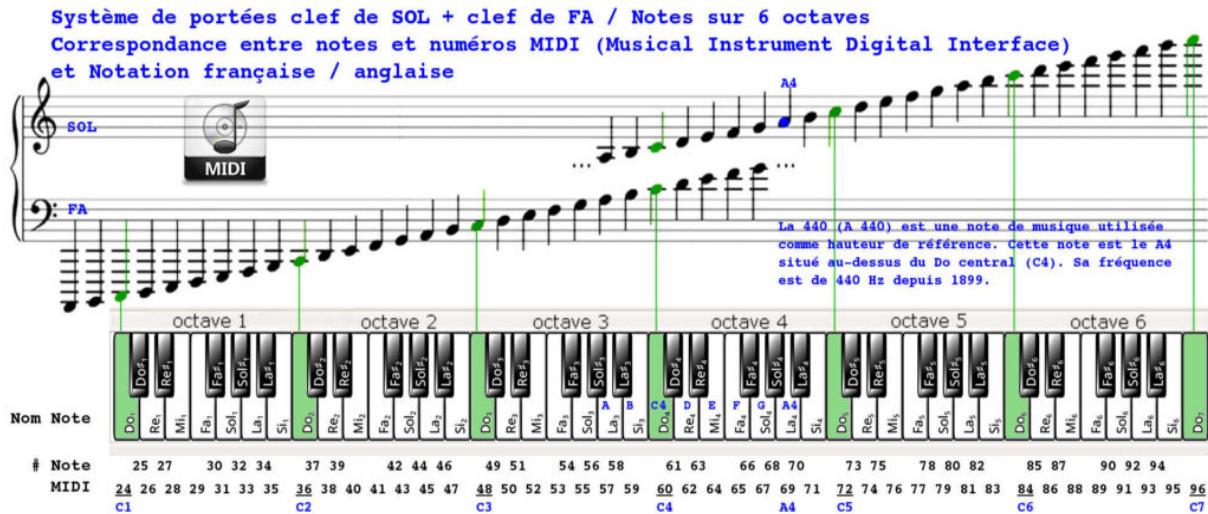


FIGURE 4 – Correspondance entre partition et note MIDI

3.1 Présentation

Le format MIDI (Musical Instrument Digital Interface) est un format de fichier musical qui permet de stocker des informations sur la musique, comme les notes jouées, leur durée, leur volume, etc. Il est utilisé par les logiciels de MAO (Musique Assistée par Ordinateur) pour enregistrer et éditer des morceaux de musique. Il est aussi utilisé par les instruments de musique électroniques pour communiquer entre eux.

En effet, le format MIDI ne contient pas de son, mais uniquement des informations sur la musique. Il est donc très léger, et permet de stocker beaucoup de musique sur un espace de stockage réduit. Il est aussi très facile à éditer, car il suffit de modifier les informations contenues dans le fichier pour modifier la musique.

3.2 Structure du fichier

Un fichier MIDI est composé de plusieurs pistes, qui contiennent chacune des événements MIDI. Un événement MIDI est une information sur la musique, comme une note jouée, un changement de volume, etc. Chaque événement MIDI est composé d'un type d'événement, et de paramètres qui dépendent du type d'événement. Par exemple, un événement MIDI de type NoteOn contient les paramètres suivants :

- Le numéro de la note jouée
 - Le numéro du canal MIDI sur lequel la note est jouée
 - La vitesse de la note (qui correspond au volume)

Un message MIDI est un message binaire composé de 3 octets, qui contient un événement MIDI. Le premier octet contient le type d'événement, et les deux autres contiennent les paramètres de l'événement. Par exemple, le message MIDI 0x90 0x3C 0x7F correspond à un événement de type **NoteOn** avec les paramètres suivants :

- Le numéro de la note jouée est 0x3C=60
 - Le numéro du canal MIDI est 0x0
 - La vitesse de la note est 0x7F

Par exemple, représentons la chanson Twinkle Twinkle Little Star à l'aide du logiciel MidiEditor :



FIGURE 5 – Logiciel libre MIDIEDITOR

Protocol	Event
Property	Value
On Tick	512
Off Tick	999
Duration	487
Note	60
Velocity	77
Channel	0

FIGURE 6 – Paramètres d'une note

On peut donc déduire des paramètres suffisants pour générer de la musique ou au moins une mélodie :

1. la note
2. la vitesse de la note
3. l'instant de départ de la note
4. la durée de la note / l'instant de fin de la note
5. L'instrument
6. La durée avant la prochaine note

3.3 Implémentation en Python

Pour l'implémentation et la syntaxe python, nous nous appuierons sur un tutoriel en ligne [Lin18].

Il a été décidé de retenir 3 paramètres définissant une note :

1. la valeur de la note (entre 0 et 127)
2. la durée de la note
3. la durée avant la prochaine note

Cela permet à deux notes de se chevaucher et de non pas juste se suivre. A l'aide de ces trois paramètres, nous pouvons ainsi restituer la musique originale d'un fichier Midi sans trop de perte d'information. La variation du volume sonore de la note jouée à dû être fixé (variation de 100% à 0%).

3.3.1 représentation en octets

```
import mido
```

```
note_on = mido.Message('note_on', note=60, velocity=112, channel=0)
print(note_on.bytes()) # Affiche [144, 60, 112]
```

3.3.2 Fonctions de passage du format MIDI au jeu de données

```
# extrait pour chaque note sa valeur, dure , attente avant prochaine note
def lireParametresMIDI(midi_file_path):
    fichierDATA = []

    with mido.MidiFile(midi_file_path) as midi_file:
        for track in midi_file.tracks:
            listeTrack = []

            i = 0
            while i < len(track):
                if
                    (track[i].type == 'note_on',
                     and
                     track[i].velocity != 0):
                        note = [track[i].note, 0, 0]
                        bool1, bool2 = True, True
                        j = i + 1

                        temps_i_j = 0

                        while
                            j < len(track)
                            and
                            (bool1 or bool2):
                                if bool1
                                    and
                                    (track[j].type == 'note_off',
                                     or
                                     (track[j].type == 'note_on',
                                      and
                                      track[j].velocity == 0))
                                    and
                                    (track[j].note == note[0]) :
                                        note[1] = temps_i_j + track[j].time
                                        bool1 = False
                                elif bool2
                                    and
                                    (track[j].type == 'note_on',
                                     and
                                     track[j].velocity != 0):
                                        note[2] = temps_i_j + track[j].time
                                        bool2 = False
                                try:
                                    temps_i_j+=track[j].time
                                except Exception as e:
                                    print("NoTime")
                                    j += 1

                        listeTrack.append(note)
                        i += 1

                fichierDATA.append(listeTrack)

    return fichierDATA
```

4 Jeu de données Maestro

Nous avons choisi de travailler avec le jeu de données en ligne Maestro [HSR⁺18].

4.1 Présentation du jeu de données

Le Dataset, structuré en format JSON, se divise en deux composantes : le fichier "maestro-v3.0.0.json" incluant les 1276 morceaux avec noms et compositeurs, accompagnés des noms des fichiers MIDI, et "DatasetParMusiques.txt" contenant les 7040164 notes des différents morceaux. Notre focus initial se porte sur "DatasetParMusiques", où chaque morceau se compose de deux pistes, une vide et l'autre contenant les notes de piano. Ces notes sont caractérisées par trois paramètres : valeur (0 à 127, d'aigu à grave), durée (en ms), et durée avant la note suivante (en ms).

4.2 Analyse statistique du dataset [Num06]

Dans cette section, nous procédons à une analyse approfondie des données, en mettant l'accent sur les trois paramètres des notes :

Présentons d'abord les moyennes de chaque paramètres :

- Valeur de note moyenne : 64.195
- Durée de note moyenne : 165.778ms
- Durée moyenne avant la prochaine note : 82.779ms
- Nombre de note moyen d'un morceau : 5517
- Durée moyenne d'un morceau : 1371387ms

4.2.1 Valeur des notes

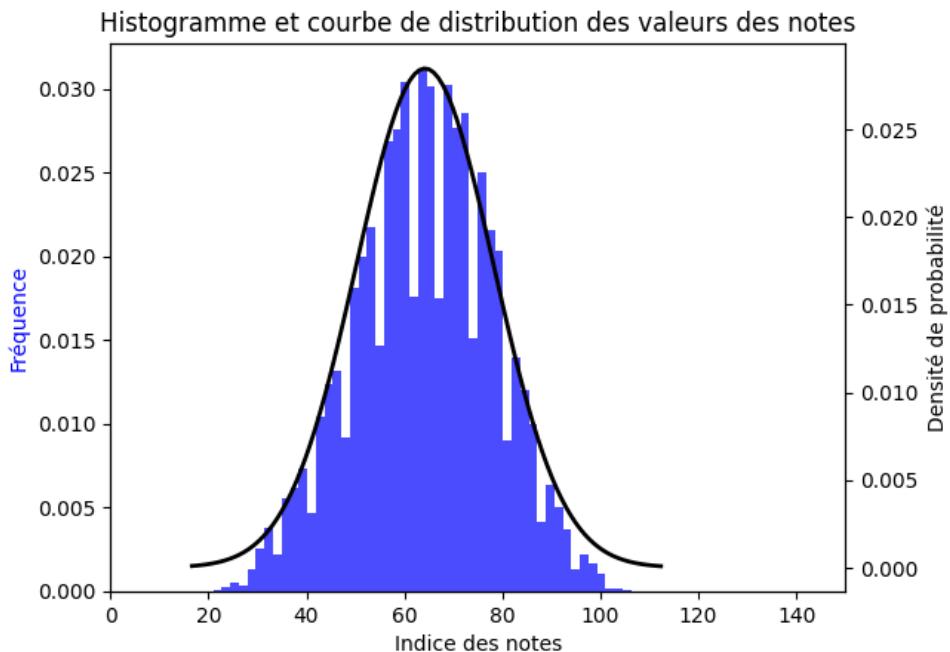


FIGURE 7 – Histogramme et courbe de distribution des valeurs des notes

L'histogramme révèle une distribution gaussienne centrée autour de la moyenne (64), indiquant une préférence pour une gamme équilibrée, typique des morceaux de piano.

Nous pouvons vérifier cette hypothèse à l'aide de différents tests :

Test de Shapiro-Wilk [DAT] :

On pose l'hypothèse selon laquelle les valeurs de notes sont distribuées selon une loi normale. En principe, si la p-value du test est inférieur au risque (ici on test à 5/100), alors l'hypothèse est invalidée, cependant pour un grand nombre de mesure (supérieur à 5000) la p-value peut ne pas être significative. On s'intéressera alors à la statistique de test, plus elle est proche de 1, moins la distribution est éloignée d'une distribution normale.

```
Test de Shapiro-Wilk pour valeurs des notes:
Statistique de test: 0.9955115914344788
P-value: 0.0
La distribution ne semble pas normale (p-value <= 0.05)
```

FIGURE 8 – Résultat du test de Shapiro-Wilk pour les valeurs des notes [Sci]

La statistique montre clairement ici que la distribution des valeurs de notes suit probablement une loi normale. On peut renforcer cette idée avec un QQ plot :

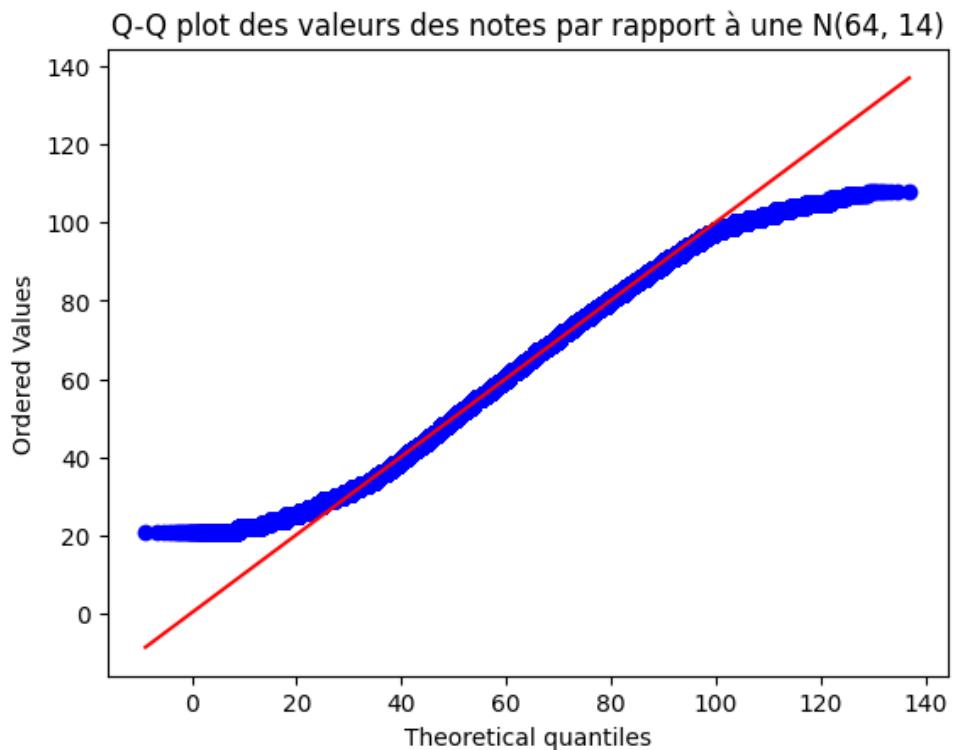


FIGURE 9 – Q-Q plot pour les valeurs des notes

Le QQ plot montre que la distribution n'est pas très éloignée d'une loi normale, on peut donc conserver notre hypothèse.

4.2.2 Durée des notes

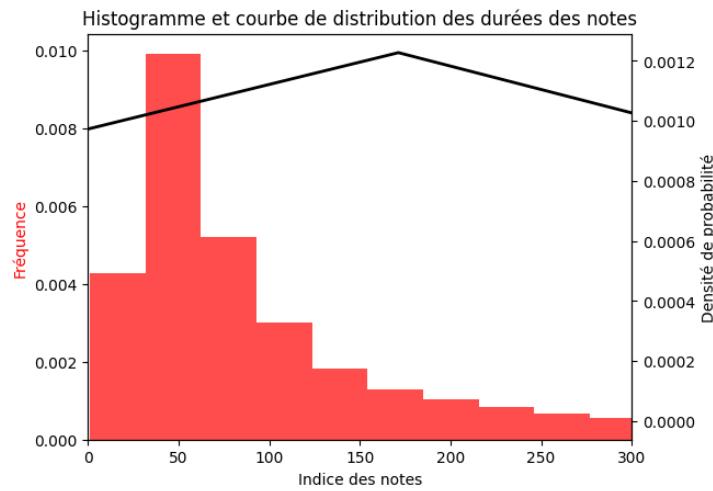


FIGURE 10 – Histogramme et courbe de distribution des durées des notes

On observe ici une distribution très orientée sur des valeurs proches de 50, les notes sont donc en grande majorité assez courte et on peut penser que l'exécution des morceaux est rapide, pour être sûr de cela, il reste à étudier le dernier paramètre.

4.2.3 Durée avant la prochaine note

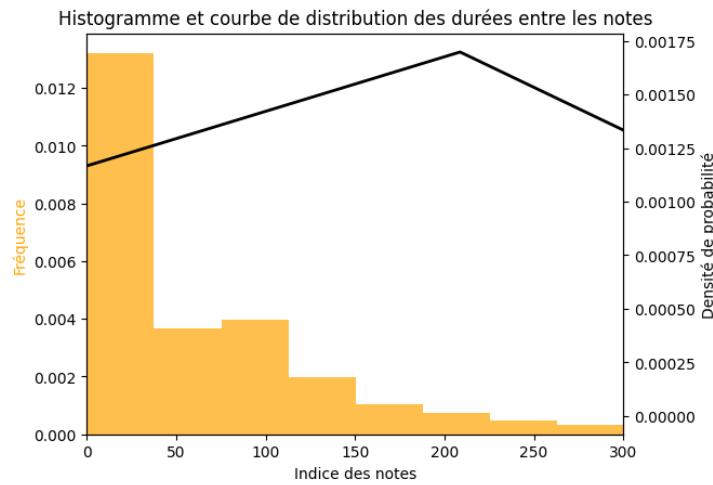


FIGURE 11 – Histogramme et courbe de distribution des durées entre les notes

Comme attendu, on observe des durées généralement courtes suggérant un enchaînement assez rapide entre les notes. Nos morceaux sont donc rythmé et centrés autour d'une game intermédiaire.

4.3 Prévision

Un autre moyen de concevoir artificiellement de la musique à partir d'une base de donnée serait de prédire directement quelle suite de note serait jouée à partir d'une note aléatoire selon la structure des morceaux du jeu de données. Nous étudions cette possibilité de production dans cette partie.

4.3.1 Prévision à une note

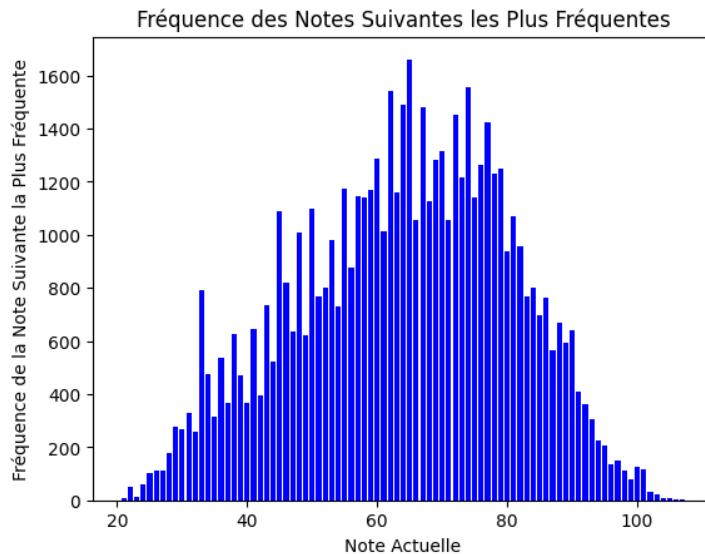


FIGURE 12 – Histogramme des notes suivantes les plus fréquentes

L'histogramme des fréquence indique une répartition normale des notes suivantes ce qui suggère que les potentiels morceaux générés par cette prévisions serait répétitifs et n'exploiterait pas réellement tout le panel des notes présent dans le dataset. Une idée renforcée par la heatmap des notes suivantes selon des paquets de 10 notes :

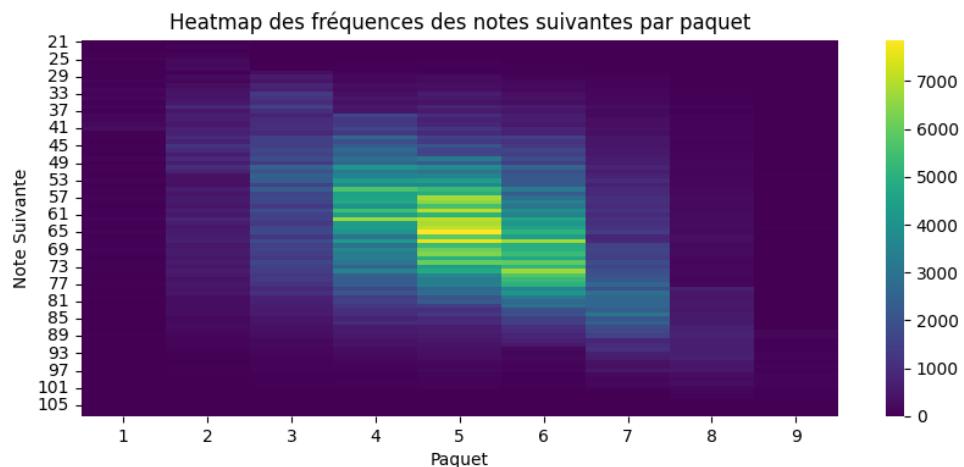


FIGURE 13 – Heatmap des notes suivantes les plus fréquentes

On voit bien que les fréquences sont fortes pour les notes centrales et faibles pour les notes extrêmes. Alors la prédiction selon une seule note ne fournirait pas de morceau musicalement riche et versatile.

4.3.2 Prévision à deux notes [Col]

Intéressons nous à la prévision à partir de doublets de notes. En relevant les notes suivantes les plus fréquentes pour chaque doublets de notes, on obtient un histogramme réparti de telle sorte qu'un morceau généré par ce type de prédiction ne serait pas redondant, cependant il y a une grande part d'aléatoire. Ainsi cette méthode ne permet pas d'obtenir de morceaux musicalement cohérent et reste largement moins performante qu'un réseau de neurones puisqu'elle est limitée, la ou un réseau de neurone peut être perfectionné.

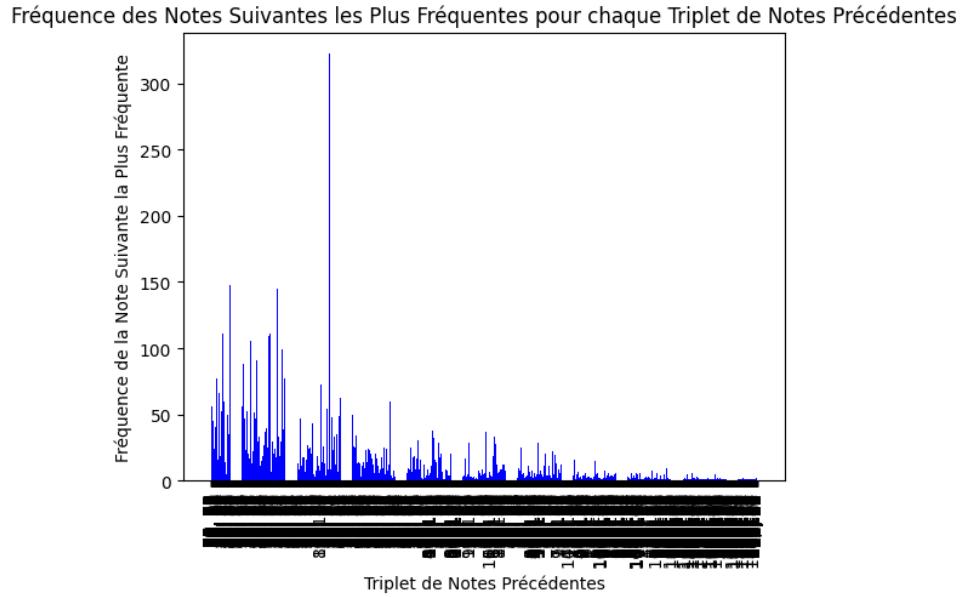


FIGURE 14 – Histogramme des notes suivantes les plus fréquentes (à partir d'un doublet de notes)

4.3.3 Prévision à plus de deux notes

En revanche, on observe rapidement qu'à plus de deux notes, le nombre de n-uplets est trop important pour obtenir un histogramme lisible. De plus les fréquences d'apparitions sont trop homogènes pour qu'elles soient supposées non aléatoire. La meilleure prévision reste celle à deux notes.

4.4 Influence des compositeurs

La dernière section se penche sur l'impact des compositeurs :

Dans notre ensemble de données, nous explorons la possibilité d'identifier des distinctions significatives dans la production musicale en fournissant à notre réseau de neurones récurrents (RNN) des morceaux provenant d'un unique compositeur et nous nous poserons la question de savoir si un seul compositeur peut refléter la tendance générale.

Nous débutons en présentant le nombre de morceaux par compositeur à l'aide d'un histogramme. Une supposition raisonnable serait que si nous fournissons un nombre restreint de morceaux à notre RNN (arbitrairement moins de 50), celui-ci pourrait ne pas être en mesure de générer une musique cohérente.

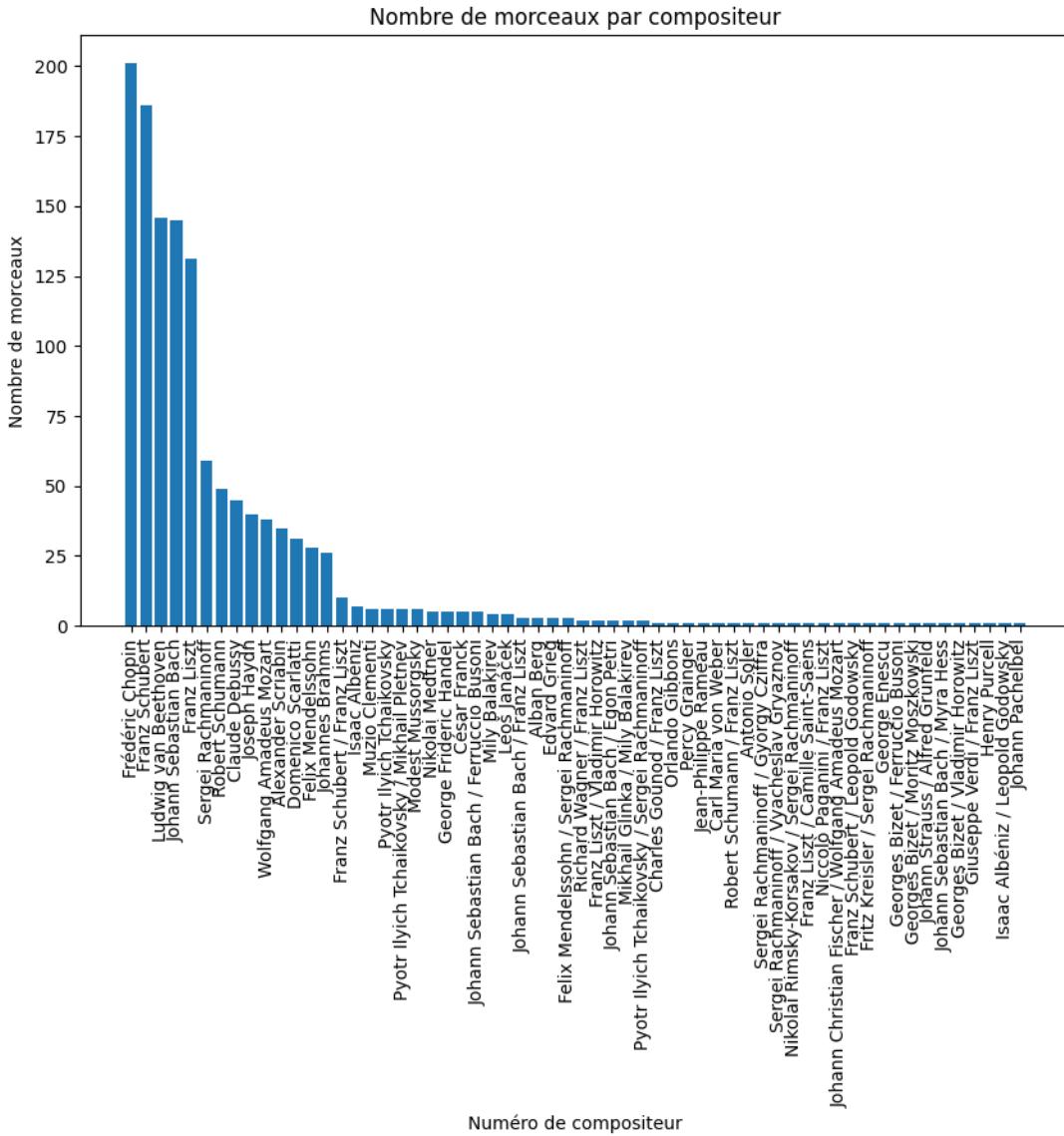


FIGURE 15 – Histogramme présentant le nombre de morceau par compositeur

Ainsi, nous retenons uniquement les compositeurs ayant plus de 50 morceaux dans le dataset, soit Chopin, Beethoven, Schubert, Liszt, Rachmaninoff et Bach. Pour chaque compositeur sélectionné, nous calculons la valeur moyenne de leurs notes, la durée moyenne des notes et la durée moyenne avant la note suivante.

Si ces valeurs se rapprochent significativement des moyennes du dataset global, nous pouvons conclure qu'un artiste peut représenter la tendance générale du dataset. En revanche, des divergences marquées suggèrent que la fourniture des morceaux de cet artiste pourrait engendrer des résultats différents, mais tout aussi captivants, avec des notes plus aiguës/graves, des durées plus courtes/longues, ou des morceaux plus rapides/lents.

```
Compositeur: Frédéric Chopin
Valeur moyenne: 64.12720111390135
Duree moyenne: 164.56030624688947
Duree entre notes moyenne: 80.9022648441032
```

FIGURE 16 – Statistiques sur les morceaux de Chopin

```
Compositeur: Ludwig van Beethoven
Valeur moyenne: 64.17595021781146
Duree moyenne: 165.50961119178203
Duree entre notes moyenne: 83.52970280643633
```

FIGURE 17 – Statistiques sur les morceaux de Beethoven

```
Compositeur: Johann Sebastian Bach
Valeur moyenne: 63.891769452743944
Duree moyenne: 175.23112694399754
Duree entre notes moyenne: 85.41276079573022
```

FIGURE 18 – Statistiques sur les morceaux de Bach

```
Compositeur: Franz Liszt
Valeur moyenne: 63.868999291579115
Duree moyenne: 169.15801696299516
Duree entre notes moyenne: 85.75058170228574
```

FIGURE 19 – Statistiques sur les morceaux de Liszt

```
Compositeur: Franz Schubert
Valeur moyenne: 64.80326971910259
Duree moyenne: 157.95507386755324
Duree entre notes moyenne: 81.71306753230887
```

FIGURE 20 – Statistiques sur les morceaux de Schubert

```
Compositeur: Sergei Rachmaninoff
Valeur moyenne: 64.10323226087388
Duree moyenne: 167.9826370706809
Duree entre notes moyenne: 82.32736287419868
```

FIGURE 21 – Statistiques sur les morceaux de Rachmaninoff

On remarque que toutes les valeurs sont relativement proches des moyennes du dataset (c'est logique puisque ces compositeurs sont les plus représentés). En revanche 1 des compositeurs semble proposer des morceaux plus longs (ou plus lent) que les autres : Bach. En effet les moyennes de durée et de durée entre les notes des morceaux de Bach sont assez éloignées de celle du dataset. Statistiquement il est facile d'observer cette divergence, mais à l'oreille ce serait probablement difficile à percevoir puisqu'il s'agit de millisecondes de différence. Néanmoins si l'on devait utiliser les musiques d'un compositeur pour obtenir un résultat cohérent musicalement et différent des morceaux générés par le dataset complet, nous utiliserions Bach.

5 Jeux de données extraits du jeu de données Maestro

Afin de poursuivre notre étude statistique et faire des essais d'entraînements de notre réseau de neurones, nous avons étudié des jeux de données tirés du dataset Maestro précédent.

5.1 Présentation des jeux de données

Le jeu de données dataset-16notes.csv comprend les 16 premières notes des musiques du jeu de données Maestro. Ce jeu de données va être utilisé pour entraîner le réseau de neurones à prédire la 16ème note en lui fournissant 15 notes. Les dataset MINI1-dataset-16notes.csv et MINI10-dataset-16notes.csv correspondent respectivement aux 1 pourcent et 10 pourcent données prises aléatoirement du dataset dataset-16notes.csv.

5.2 Jeu de données MINI1-dataset

Ce jeu de données correspond aux 1 pourcent notes aléatoirement issues du dataset dataset-16notes.csv. Une note est ici représentées par ses trois composantes (valeur,durée et durée avant la prochaine note) notées respectivement note-X-1,note-X-2 et note-X-3 où X est le numéro de la note. Nous obtenons les résultats statistiques suivants :

	note_1_1	note_1_2	note_1_3	note_2_1	note_2_2 \
count	15608.000000	15608.000000	15608.000000	15608.000000	15608.000000
mean	64.132817	158.560290	83.163250	64.200282	161.539083
std	14.037341	275.087141	194.589374	14.025761	279.876405
min	24.000000	1.000000	0.000000	21.000000	1.000000
25%	55.000000	41.000000	6.000000	55.000000	41.000000
50%	64.000000	72.000000	40.000000	65.000000	73.000000
75%	74.000000	158.000000	101.250000	74.000000	159.000000
max	104.000000	5436.000000	8027.000000	104.000000	6615.000000
	note_2_3	note_3_1	note_3_2	note_3_3	note_4_1 \
count	15608.000000	15608.000000	15608.000000	15608.000000	15608.000000
mean	82.868145	64.180292	164.516850	81.344887	63.998206
std	203.292474	13.949980	307.295532	163.588248	14.184940
min	0.000000	21.000000	1.000000	0.000000	22.000000
25%	6.000000	55.000000	41.000000	6.000000	54.000000
50%	39.000000	64.000000	73.000000	39.000000	64.000000
75%	102.000000	74.000000	159.000000	102.250000	74.000000
max	10885.000000	106.000000	10393.000000	9459.000000	103.000000
	note_4_2	note_4_3	note_5_1	note_5_2	note_5_3 \
count	15608.000000	15608.000000	15608.000000	15608.000000	15608.000000
mean	164.135315	81.484367	64.302281	164.425231	82.006087
std	311.128105	147.476199	14.156097	319.386987	167.910277
min	1.000000	0.000000	21.000000	1.000000	0.000000
25%	41.000000	6.000000	55.000000	41.000000	6.000000
50%	73.000000	39.000000	65.000000	73.000000	37.000000
75%	158.000000	102.250000	74.000000	159.000000	102.000000
max	10657.000000	3326.000000	103.000000	10482.000000	5277.000000
	note_6_1	note_6_2	note_6_3	note_7_1	note_7_2 \
count	15608.000000	15608.000000	15608.000000	15608.000000	15608.000000
mean	64.069516	162.731035	80.758393	64.342068	163.588544
std	14.030625	322.574447	154.770776	13.998244	328.501533
min	21.000000	1.000000	0.000000	21.000000	1.000000
25%	55.000000	41.000000	6.000000	55.000000	41.000000
50%	64.000000	73.000000	39.000000	65.000000	73.000000
75%	74.000000	158.000000	101.000000	74.000000	158.000000
max	105.000000	10680.000000	5929.000000	104.000000	10531.000000
	note_7_3	note_8_1	note_8_2	note_8_3	note_9_1 \
count	15608.000000	15608.000000	15608.000000	15608.000000	15608.000000
mean	81.634610	64.108983	159.941184	81.628524	64.139031
std	174.048634	13.994424	310.329165	161.857626	14.018615
min	0.000000	21.000000	1.000000	0.000000	22.000000
25%	6.000000	55.000000	41.000000	6.000000	55.000000
50%	39.000000	64.000000	72.000000	39.000000	64.000000
75%	102.000000	74.000000	155.000000	102.000000	74.000000
max	7815.000000	105.000000	10294.000000	6629.000000	108.000000

FIGURE 22 – Statistiques descriptives MINI1

	note_9_2	note_9_3	note_10_1	note_10_2	note_10_3
count	15608.000000	15608.000000	15608.000000	15608.000000	15608.000000
mean	162.360136	80.501922	64.249936	163.662801	85.193619
std	295.711492	162.071808	13.989341	309.980207	282.899197
min	1.000000	0.000000	21.000000	1.000000	0.000000
25%	41.000000	6.000000	55.000000	41.000000	6.000000
50%	72.000000	38.000000	65.000000	73.000000	38.000000
75%	160.000000	101.000000	74.000000	161.000000	101.250000
max	8021.000000	6796.000000	105.000000	8090.000000	23694.000000
	note_11_1	note_11_2	note_11_3	note_12_1	note_12_2 \
count	15608.000000	15608.000000	15608.000000	15608.000000	15608.000000
mean	64.22213	159.850654	82.083547	64.172027	165.302858
std	13.87770	285.668989	167.424396	13.945718	309.097044
min	22.000000	1.000000	0.000000	21.000000	1.000000
25%	55.000000	40.000000	6.000000	55.000000	41.000000
50%	64.000000	71.000000	38.000000	64.000000	72.000000
75%	74.000000	156.000000	101.000000	74.000000	158.000000
max	105.000000	6251.000000	6033.000000	108.000000	7688.000000
	note_12_3	note_13_1	note_13_2	note_13_3	note_14_1
count	15608.000000	15608.000000	15608.000000	15608.000000	15608.000000
mean	82.639095	64.076948	168.595015	81.065159	64.079190
std	193.047624	13.951353	329.842728	159.498282	13.999282
min	0.000000	21.000000	1.000000	0.000000	21.000000
25%	6.000000	55.000000	41.000000	6.000000	55.000000
50%	38.000000	64.000000	73.000000	38.000000	64.000000
75%	102.000000	74.000000	162.000000	102.000000	74.000000
max	13449.000000	106.000000	8236.000000	8325.000000	105.000000
	note_14_2	note_14_3	note_15_1	note_15_2	note_15_3
count	15608.000000	15608.000000	15608.000000	15608.000000	15608.000000
mean	168.399987	84.700474	64.138647	165.907868	81.169080
std	327.767106	188.203006	14.017138	343.415093	157.397076
min	1.000000	0.000000	21.000000	1.000000	0.000000
25%	41.000000	6.000000	55.000000	41.000000	6.000000
50%	72.000000	40.000000	64.000000	73.000000	37.000000
75%	160.000000	103.000000	74.000000	159.000000	102.000000
max	6685.000000	8708.000000	104.000000	15411.000000	5212.000000
	note_16_1	note_16_2	note_16_3		
count	15608.000000	15608.000000	15608.000000		
mean	64.187212	163.187532	84.319195		
std	13.956965	311.083569	254.690612		
min	22.000000	1.000000	0.000000		
25%	55.000000	41.000000	6.000000		
50%	64.000000	72.000000	39.000000		
75%	74.000000	159.000000	102.000000		
max	106.000000	7605.000000	18821.000000		

FIGURE 23 – Statistiques descriptives MINI1 suite

Nous observons ici une liste de statistiques descriptives pour chaque caractéristique de chaque note. On remarque que sur l'ensemble des notes, les statistiques concernant une caractéristique sont très proches entre elles, pouvant indiquer des répartitions de chaque caractéristique très similaires entre les notes. Pour appuyer nos propos, affichons les histogrammes de chaque caractéristique pour chaque note :

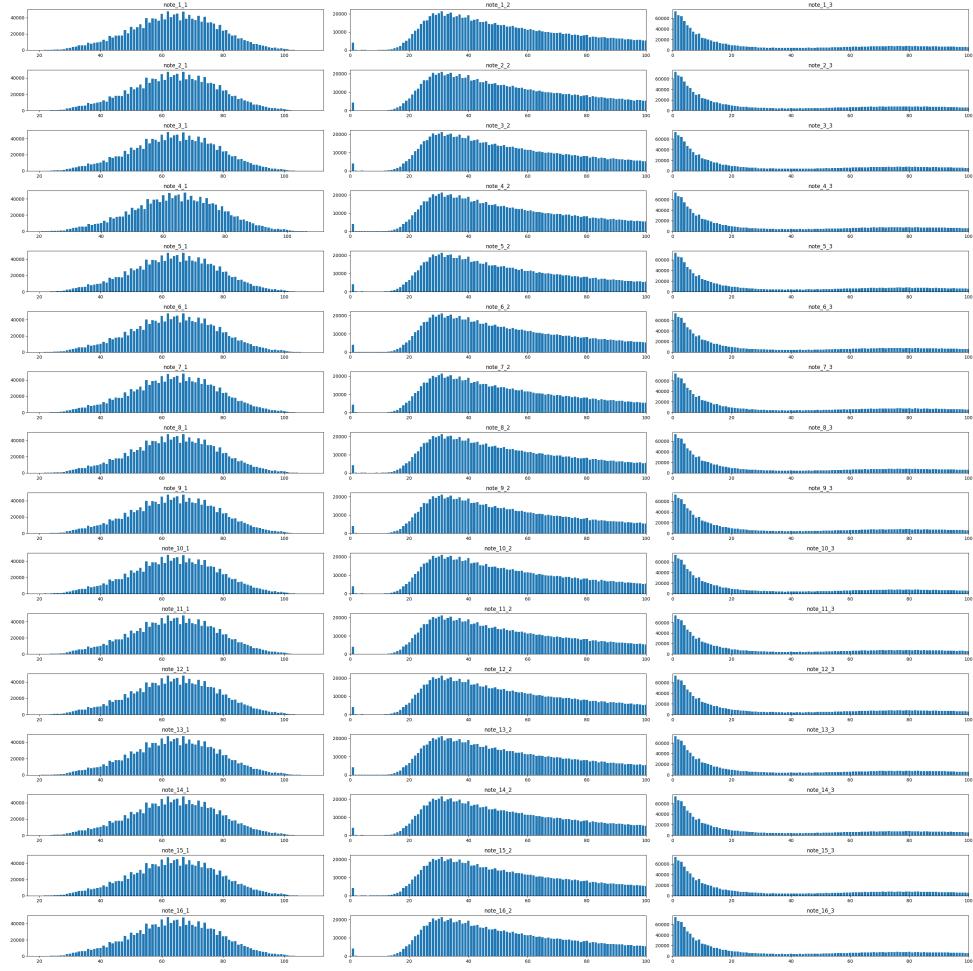


FIGURE 24 – Histogramme MINI1

Ici, on remarque effectivement que chaque caractéristique de la note a une distribution très similaire d'une note à l'autre. Ceci est avantageux car va permettre au réseau de neurones de ne pas privilégier une note par rapport à une autre lors de son entraînement.

Avant de passer aux étapes d'apprentissage, de test et de prédiction, il est nécessaire de vérifier si toutes les variables de notre jeu de données (ici, les caractéristiques de chaque note) sont non-correlées et éliminer celles qui sont très fortement corrélées pour éviter de la redondance et une non-significativité de certains scores de précision. Nous avons alors produit une matrice de corrélation sous la forme d'une heatmap, étant donné le nombre important de variables du dataset, pour mieux visualiser les choses. Une couleur sombre signifie un coefficient de corrélation proche de 0 et une couleur qui s'approche du blanc signifie un coefficient de corrélation proche de 1. Voici la matrice obtenue :

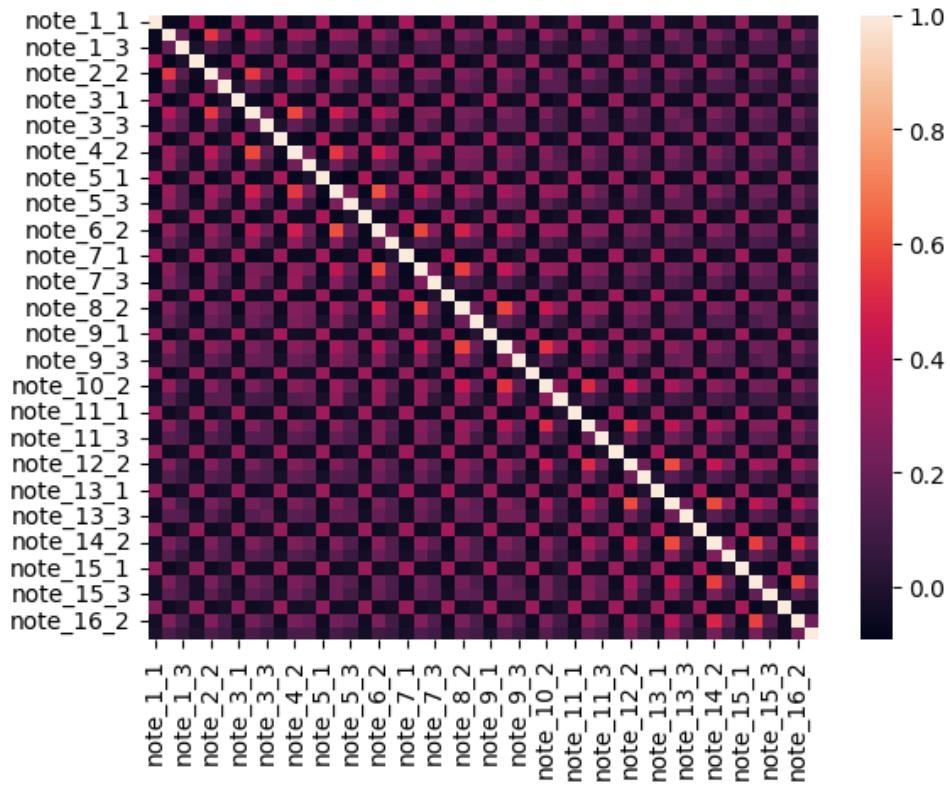


FIGURE 25 – Matrice de corrélation heatmap MINI1

Nous remarquons ici que la diagonale de la matrice a une couleur blanche, ce qui est normal car il s'agit ici des coefficients de corrélation de la variable avec elle-même (ce qui est égal à 1). Les autres couleurs sont toutes sombres, voire au maximum rouge (coefficient de corrélation d'environ 0.5) et nous ne considérons donc pas les variables fortement corrélées entre elles. Nous n'avons donc pas à éliminer une variable.

5.3 Jeu de données MINI10-dataset

Nous obtenons des résultats similaires aux résultats sur le jeu de données 1 pourcent :

	note_1_1	note_1_2	note_1_3	note_2_1	\
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	64.243146	162.555972	82.860507	64.228366	
std	14.035792	303.848689	195.217843	14.037344	
min	21.000000	1.000000	0.000000	21.000000	
25%	55.000000	41.000000	6.000000	55.000000	
50%	65.000000	72.000000	39.000000	65.000000	
75%	74.000000	157.000000	101.000000	74.000000	
max	106.000000	12665.000000	15853.000000	107.000000	
	note_2_2	note_2_3	note_3_1	note_3_2	\
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	162.981632	82.278281	64.236400	163.864998	
std	312.562181	182.889285	14.025793	321.242179	
min	1.000000	0.000000	21.000000	1.000000	
25%	41.000000	6.000000	55.000000	41.000000	
50%	72.000000	40.000000	64.000000	73.000000	
75%	156.000000	102.000000	74.000000	158.000000	
max	12335.000000	18570.000000	107.000000	29981.000000	
	note_3_3	note_4_1	note_4_2	note_4_3	\
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	82.659767	64.167451	163.216213	82.100950	
std	175.214470	14.034482	317.584594	198.663939	
min	0.000000	21.000000	1.000000	0.000000	
25%	6.000000	55.000000	41.000000	6.000000	
50%	39.000000	64.000000	72.000000	38.000000	
75%	102.000000	74.000000	157.000000	101.000000	
max	11808.000000	106.000000	29850.000000	20663.000000	
	note_5_1	note_5_2	note_5_3	note_6_1	\
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	64.242685	162.842844	82.605111	64.225509	
std	13.991419	317.753515	180.200268	14.025765	
min	21.000000	1.000000	0.000000	21.000000	
25%	55.000000	41.000000	6.000000	55.000000	
50%	64.000000	72.000000	39.000000	65.000000	
75%	74.000000	157.000000	102.000000	74.000000	
max	106.000000	29101.000000	14967.000000	108.000000	
	note_6_2	note_6_3	note_7_1	note_7_2	\
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	163.282996	82.713013	64.252058	163.125808	
std	325.757635	206.191247	14.008570	321.430990	
min	1.000000	0.000000	21.000000	1.000000	
25%	41.000000	6.000000	55.000000	41.000000	
50%	73.000000	40.000000	65.000000	72.000000	
75%	157.000000	102.000000	74.000000	157.000000	
max	30648.000000	25136.000000	108.000000	30248.000000	
	note_7_3	note_8_1	note_8_2	note_8_3	\
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	81.820344	64.230134	162.823208	82.573469	
std	174.723883	14.033617	318.246479	213.258935	
min	0.000000	21.000000	1.000000	0.000000	
25%	6.000000	55.000000	41.000000	6.000000	
50%	39.000000	64.000000	72.000000	38.000000	
75%	102.000000	74.000000	157.000000	101.000000	
max	14620.000000	108.000000	30333.000000	30796.000000	

FIGURE 26 – Statistiques descriptives MINI10

	note_9_1	note_9_2	note_9_3	note_10_1	\
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	64.182270	162.998161	82.082800	64.251898	
std	14.041136	302.385096	179.612499	14.007037	
min	21.000000	1.000000	0.000000	21.000000	
25%	55.000000	41.000000	6.000000	55.000000	
50%	64.000000	72.000000	39.000000	64.000000	
75%	74.000000	157.000000	101.000000	74.000000	
max	108.000000	9134.000000	20959.000000	107.000000	
	note_10_2	note_10_3	note_11_1	note_11_2	\
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	162.980601	82.627093	64.211036	163.131875	
std	304.010462	205.078616	13.999972	306.095722	
min	1.000000	0.000000	21.000000	1.000000	
25%	41.000000	6.000000	55.000000	41.000000	
50%	72.000000	39.000000	64.000000	72.000000	
75%	158.000000	101.000000	74.000000	158.000000	
max	11402.000000	23694.000000	106.000000	11928.000000	
	note_11_3	note_12_1	note_12_2	note_12_3	\
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	82.020162	64.214195	163.166792	82.408343	
std	173.515068	13.9990725	308.664313	171.125354	
min	0.000000	21.000000	1.000000	0.000000	
25%	6.000000	55.000000	41.000000	6.000000	
50%	39.000000	64.000000	72.000000	39.000000	
75%	101.000000	74.000000	158.000000	102.000000	
max	13860.000000	108.000000	12656.000000	13449.000000	
	note_13_1	note_13_2	note_13_3	note_14_1	\
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	64.215386	163.947529	82.618296	64.206929	
std	13.969250	317.804371	202.236423	13.982967	
min	21.000000	1.000000	0.000000	21.000000	
25%	55.000000	41.000000	6.000000	55.000000	
50%	64.000000	73.000000	38.000000	64.000000	
75%	74.000000	158.000000	101.500000	74.000000	
max	108.000000	13710.000000	30796.000000	108.000000	
	note_14_2	note_14_3	note_15_1	note_15_2	\
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	164.385324	81.983368	64.220159	164.353739	
std	322.844016	180.679118	14.003839	324.956498	
min	1.000000	0.000000	21.000000	1.000000	
25%	41.000000	6.000000	55.000000	41.000000	
50%	72.000000	39.000000	64.000000	73.000000	
75%	159.000000	101.000000	74.000000	158.000000	
max	15056.000000	16520.000000	106.000000	15411.000000	
	note_15_3	note_16_1	note_16_2	note_16_3	
count	156087.000000	156087.000000	156087.000000	156087.000000	
mean	82.055905	64.214521	164.449480	82.417857	
std	185.213810	14.011547	326.309433	206.518694	
min	0.000000	21.000000	1.000000	0.000000	
25%	6.000000	55.000000	41.000000	6.000000	
50%	39.000000	64.000000	72.000000	39.000000	
75%	102.000000	74.000000	157.000000	101.000000	
max	28682.000000	108.000000	13314.000000	33010.000000	

FIGURE 27 – Statistiques descriptives MINI10 suite

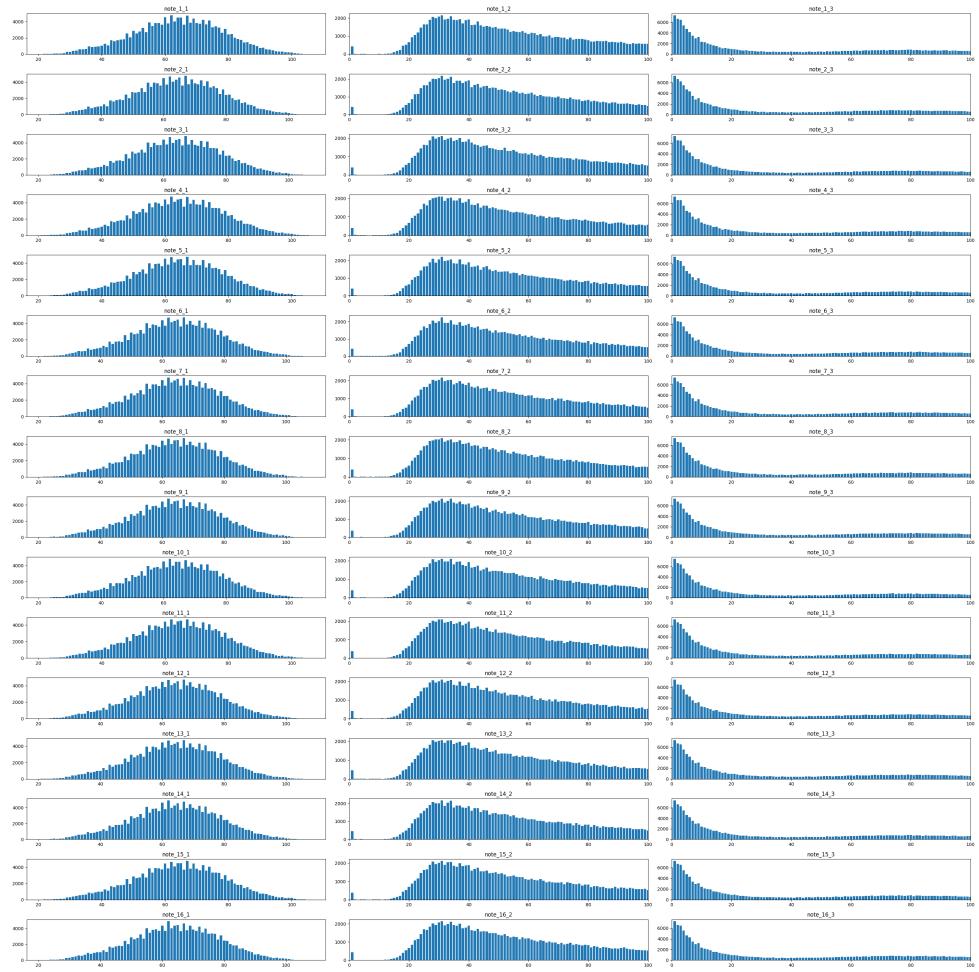


FIGURE 28 – Histogramme MINI10

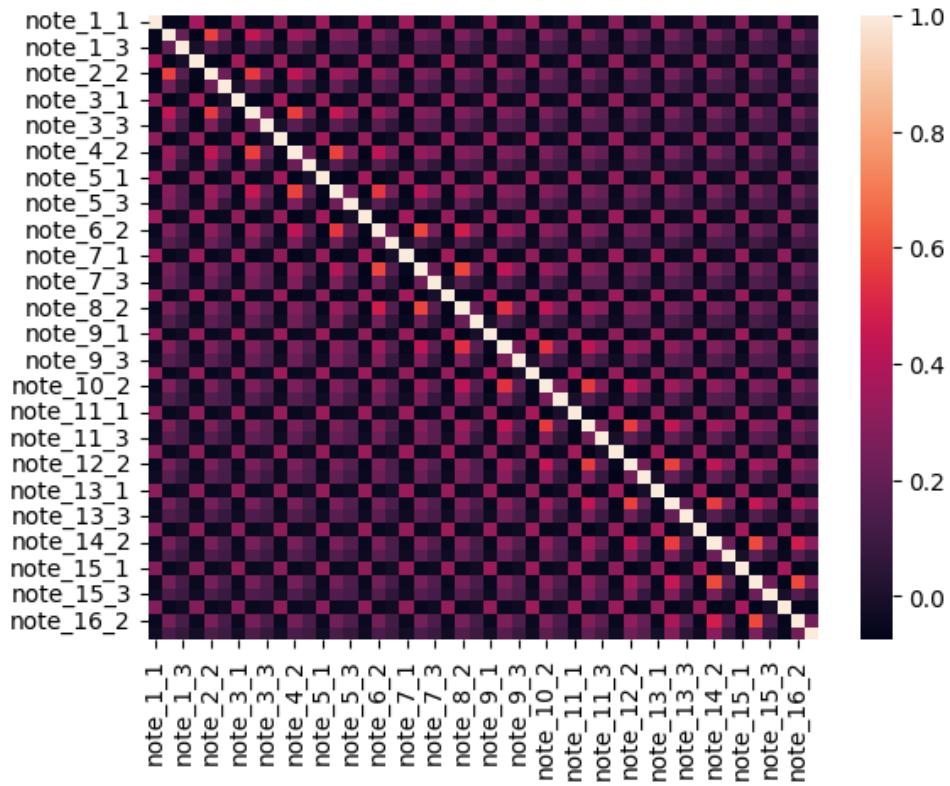


FIGURE 29 – heatmap MINI10

5.4 Jeu de données complet dataset-16notes

Les résultats obtenus sur le jeu de données sont très proches de ceux obtenus sur les jeux de données réduits : les statistiques descriptives varient peu malgré le grand nombre de données et les histogrammes sont plus denses mais similaires aux précédents :

	note_1_1	note_1_2	note_1_3	note_2_1	note_2_2	\
count	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	
mean	6.421247e+01	1.640009e+02	8.259448e+01	6.420427e+01	1.640767e+02	
std	1.398947e+01	3.170977e+02	1.906483e+02	1.399991e+01	3.170379e+02	
min	2.100000e+01	1.000000e+00	0.000000e+00	2.100000e+01	1.000000e+00	
25%	5.500000e+01	4.100000e+01	6.000000e+00	5.500000e+01	4.100000e+01	
50%	6.400000e+01	7.300000e+01	3.900000e+01	6.400000e+01	7.300000e+01	
75%	7.400000e+01	1.580000e+02	1.020000e+02	7.400000e+01	1.580000e+02	
max	1.080000e+02	3.064800e+04	3.756800e+04	1.080000e+02	3.033300e+04	
	note_2_3	note_3_1	note_3_2	note_3_3	note_4_1	\
count	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	
mean	8.250517e+01	6.419411e+01	1.641440e+02	8.253056e+01	6.420898e+01	
std	1.903411e+02	1.399699e+01	3.160246e+02	1.772006e+02	1.399525e+01	
min	0.000000e+00	2.100000e+01	1.000000e+00	0.000000e+00	2.100000e+01	
25%	6.000000e+00	5.500000e+01	4.100000e+01	6.000000e+00	5.500000e+01	
50%	3.900000e+01	6.400000e+01	7.300000e+01	3.900000e+01	6.400000e+01	
75%	1.020000e+02	7.400000e+01	1.580000e+02	1.020000e+02	7.400000e+01	
max	3.079600e+04	1.080000e+02	3.033300e+04	2.227400e+04	1.070000e+02	
	note_4_2	note_4_3	note_5_1	note_5_2	note_5_3	\
count	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	
mean	1.636496e+02	8.257884e+01	6.422178e+01	1.638075e+02	8.246373e+01	
std	3.135005e+02	1.885521e+02	1.398410e+01	3.134404e+02	1.833403e+02	
min	1.000000e+00	0.000000e+00	2.100000e+01	1.000000e+00	0.000000e+00	
25%	4.100000e+01	6.000000e+00	5.500000e+01	4.100000e+01	6.000000e+00	
50%	7.300000e+01	3.900000e+01	6.400000e+01	7.300000e+01	3.900000e+01	
75%	1.580000e+02	1.020000e+02	7.400000e+01	1.590000e+02	1.020000e+02	
max	2.985000e+04	3.284400e+04	1.080000e+02	2.910100e+04	2.681300e+04	
	note_6_1	note_6_2	note_6_3	note_7_1	note_7_2	\
count	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	
mean	6.421118e+01	1.640523e+02	8.239770e+01	6.420110e+01	1.635444e+02	
std	1.399219e+01	3.151725e+02	1.836871e+02	1.398982e+01	3.111135e+02	
min	2.100000e+01	1.000000e+00	0.000000e+00	2.100000e+01	1.000000e+00	
25%	5.500000e+01	4.100000e+01	6.000000e+00	5.500000e+01	4.100000e+01	
50%	6.400000e+01	7.300000e+01	3.900000e+01	6.400000e+01	7.300000e+01	
75%	7.400000e+01	1.590000e+02	1.020000e+02	7.400000e+01	1.580000e+02	
max	1.080000e+02	3.064800e+04	2.621300e+04	1.080000e+02	3.024800e+04	

FIGURE 30 – Statistiques descriptives du dataset complet

	note_9_2	note_9_3	note_10_1	note_10_2	note_10_3	\
count	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	
mean	1.635007e+02	8.233597e+01	6.421324e+01	1.635961e+02	8.256922e+01	
std	3.103726e+02	1.790148e+02	1.398997e+01	3.110916e+02	1.912664e+02	
min	1.000000e+00	0.000000e+00	2.100000e+01	1.000000e+00	0.000000e+00	
25%	4.100000e+01	6.000000e+00	5.500000e+01	4.100000e+01	6.000000e+00	
50%	7.300000e+01	3.900000e+01	6.480000e+01	7.300000e+01	3.900000e+01	
75%	1.580000e+02	1.020000e+02	7.400000e+01	1.580000e+02	1.020000e+02	
max	2.469300e+04	3.071700e+04	1.080000e+02	2.998100e+04	2.927800e+04	
	note_11_1	note_11_2	note_11_3	note_12_1	note_12_2	\
count	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	
mean	6.420939e+01	1.641942e+02	8.236174e+01	6.420534e+01	1.643365e+02	
std	1.400046e+01	3.143152e+02	1.762520e+02	1.400228e+01	3.159920e+02	
min	2.100000e+01	1.000000e+00	0.000000e+00	2.100000e+01	1.000000e+00	
25%	5.500000e+01	4.100000e+01	6.000000e+00	5.500000e+01	4.100000e+01	
50%	6.400000e+01	7.300000e+01	3.900000e+01	6.400000e+01	7.300000e+01	
75%	7.400000e+01	1.590000e+02	1.020000e+02	7.400000e+01	1.580000e+02	
max	1.080000e+02	2.985000e+04	1.996900e+04	1.080000e+02	2.910100e+04	
	note_12_3	note_13_1	note_13_2	note_13_3	note_14_1	\
count	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	
mean	8.262474e+01	6.420460e+01	1.644774e+02	8.235633e+01	6.420539e+01	
std	1.898681e+02	1.399571e+01	3.186337e+02	1.860157e+02	1.399868e+01	
min	0.000000e+00	2.100000e+01	1.000000e+00	0.000000e+00	2.100000e+01	
25%	6.000000e+00	5.500000e+01	4.100000e+01	6.000000e+00	5.500000e+01	
50%	3.900000e+01	6.400000e+01	7.300000e+01	3.900000e+01	6.400000e+01	
75%	1.020000e+02	7.400000e+01	1.590000e+02	1.020000e+02	7.400000e+01	
max	3.756800e+04	1.080000e+02	3.064800e+04	3.284480e+04	1.080000e+02	
	note_14_2	note_14_3	note_15_1	note_15_2	note_15_3	\
count	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	1.560879e+06	
mean	1.649491e+02	8.265383e+01	6.418718e+01	1.647516e+02	8.268050e+01	
std	3.201544e+02	1.876483e+02	1.400659e+01	3.217602e+02	1.881413e+02	
min	1.000000e+00	0.000000e+00	2.100000e+01	1.000000e+00	0.000000e+00	
25%	4.100000e+01	6.000000e+00	5.500000e+01	4.100000e+01	6.000000e+00	
50%	7.300000e+01	3.900000e+01	6.400000e+01	7.300000e+01	3.900000e+01	
75%	1.590000e+02	1.020000e+02	7.400000e+01	1.580000e+02	1.020000e+02	
max	3.024800e+04	3.071700e+04	1.080000e+02	3.033300e+04	2.868200e+04	
	note_16_1	note_16_2	note_16_3			
count	1.560879e+06	1.560879e+06	1.560879e+06			
mean	6.420737e+01	1.652091e+02	8.252744e+01			
std	1.400890e+01	3.224626e+02	1.852954e+02			
min	2.100000e+01	1.000000e+00	0.000000e+00			
25%	5.500000e+01	4.100000e+01	6.000000e+00			
50%	6.400000e+01	7.300000e+01	3.900000e+01			
75%	7.400000e+01	1.580000e+02	1.020000e+02			
max	1.080000e+02	2.998100e+04	3.301000e+04			

FIGURE 31 – Statistiques descriptives du dataset complet (suite)

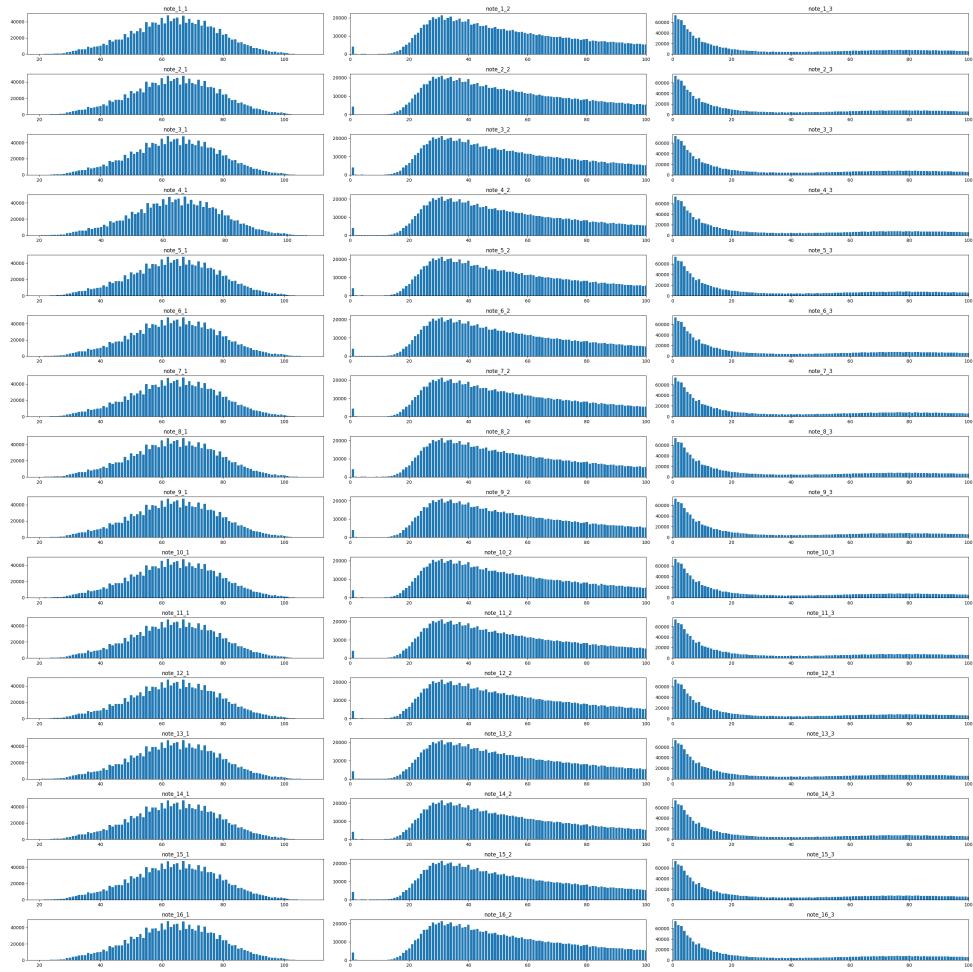


FIGURE 32 – histogrammes de chaque variable du dataset complet

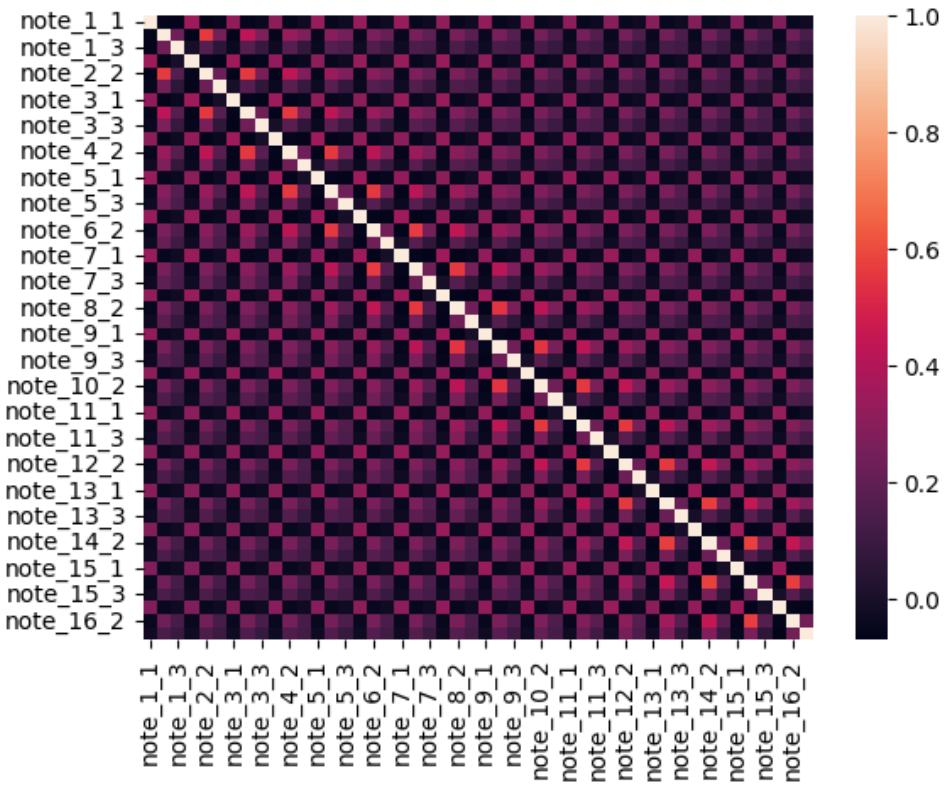


FIGURE 33 – Matrice des corrélations en heatmap du dataset complet

5.5 Conclusions à tirer

La première chose que l'on peut conclure est que toutes les notes sont représentées de la même manière, que ce soit en valeur, en durée ou en temps d'attente avant la prochaine note. Egalement, il est important de noter que les 16èmes notes (celles à prédire) sont aussi distribuées de la même manière que les autres notes. Finalement, nous pouvons noter que pour des études statistiques ultérieures, les résultats statistiques sur les échantillons 1 et 10 pourront être représentatifs, nous pouvons nous limiter à ceux-ci pour des études statistiques ultérieures au lieu de faire des calculs sur la base de données complète, limitant ainsi les temps de calculs.

6 Entraînement du réseau de neurones

Afin de "créer" de la musique avec un réseau LSTM, il faut apprendre au réseau à prédire les M notes qui suivent N-M notes dans une séquence de N notes. Ainsi, on donne en entrée N-M notes au réseau de neurones, qui va fournir M notes en sortie. Ces M notes vont s'ajouter aux N-M notes, et on redonne indéfiniment les N-M dernières notes générées.

Nous avons décidé de prendre une séquence de 16 notes, et d'avoir en entrée les 15 premières notes pour prédire la 16e. Donc $N = 16$ et $M = 1$.

La musique se décomposant en 8 temps, prendre 16 notes nous semblaient pertinent, surtout quand dans des modèles précédemment créés, des notes se répétaient assez longtemps.

Hypothétiquement, prendre 16 notes (valeur, durée, durée avant prochaine note) permet donc au réseau de comprendre à quel moment des 8 temps il se situe et donc de ne pas se répéter indéfiniment.

L'un de nos enjeux est de nous en assurer.

6.1 Les données

Nos données sont des séquences de 16 notes, extraites aléatoirement dans l'ensemble des fichiers .midi du jeu de données MAESTRO.

```
#Extrait du fichier G.DATA.py
...
i=0
while i+longueurSequence<=tailleListe :
    res.append(
        [ listeNotes [ i : i+longueurSequence ] [ j ][ k ]
        ... for j in range(longueurSequence)
        ... for k in range(3) ]
    )
    i += random.randint(1, longueurSequence // 2)
...
```

Donc les 16 toutes premières notes de chaque morceau sont présentes dans le jeu d'entraînement/validation. Cela permet par exemple de comparer le morceau original d'un morceau généré par le réseau LSTM. En effet, il faut fournir les 15 premières notes au réseau LSTM et on peut donc observer à quel point nos prédictions diverge du morceau original (qu'il soit dans le jeu d'entraînement ou non).

6.2 Choix du modèle

Il a été décidé d'entraîner un réseau de neurones LSTM, et d'observer l'influence de différents paramètres.

D'abord des paramètres fixes :

1. Le nombre de Batch, ici 32. Donc pour N données, il y a $\frac{N}{32}$ données traitées à chaque mise à jour du modèle.
2. Early Stopping de 5 époques. Donc si au bout de 5 étapes il n'y a pas de nette amélioration, l'entraînement s'arrête.

Puis des paramètres variables :

1. Le Batch Normalization entre la sortie de la couche LSTM et la couche dense de la prédiction. (Oui : 1, Non : 0)
2. Dropout entre la sortie de la couche LSTM et la couche dense de la prédiction. On a un Dropout fixe de 20%. (Oui : 1, Non : 0)
3. Le nombre de neurones : 20 neurones puis 100 neurones dans un premier temps.
4. Le pourcentage du jeu de données MAESTRO : 1%, 10% et 100% des morceaux de piano.

On obtient le tableau suivant :

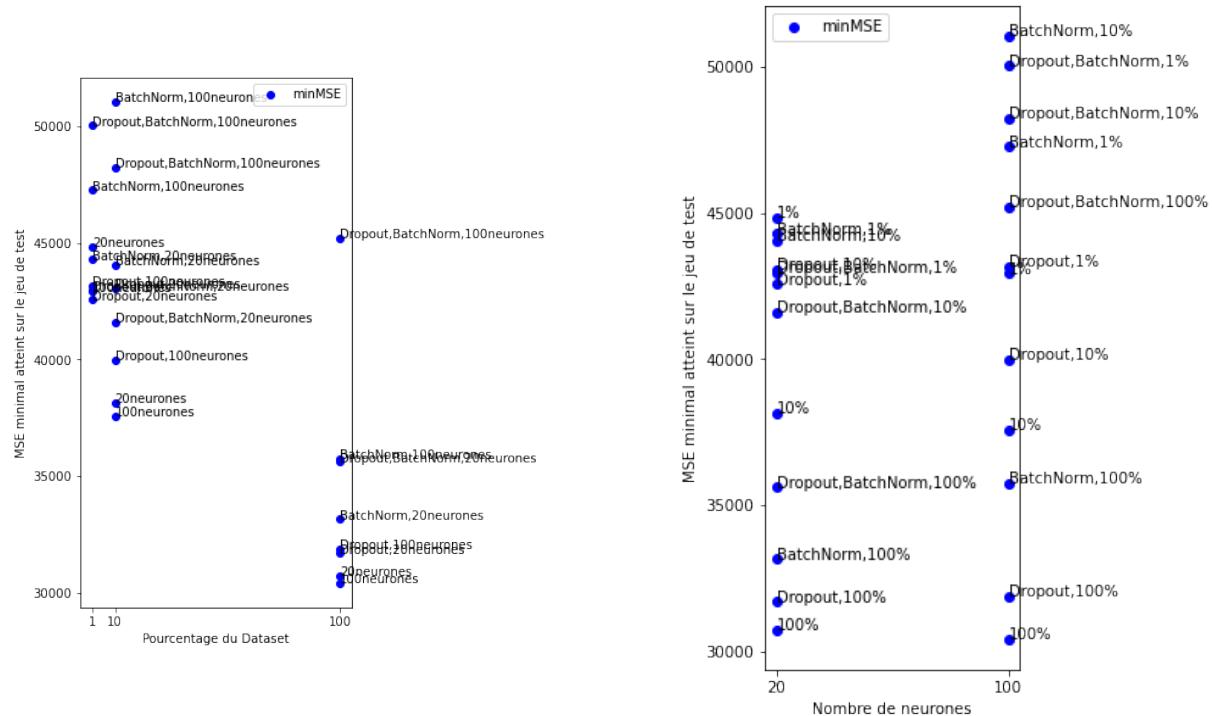
Pourcentage du dataset	nombre de neurones	Dropout	batch Normalization	epoch	minMSE	Numéro de photo
1	20	0	1	1	17 44290.609375	1
1	20	1	0	0	27 44290.59765625	2
1	20	1	1	1	13 42573.31640625	3
1	100	0	0	0	42 42944.87109375	4
1	100	0	1	1	8 42927.717875	5
1	100	1	0	0	15 47274.40625	6
1	100	1	1	1	12 43183.86328125	7
1	100	1	1	1	9 50033.05078125	8
10	20	0	0	0	16 38133.5703125	9
10	20	0	1	1	14 44072.8046875	10
10	20	1	0	0	7 43057.47265625	11
10	20	1	1	1	16 41599.72265625	12
10	100	0	0	0	19 37595.12109375	13
10	100	0	1	1	7 51067.93359375	14
10	100	1	0	0	7 39891.2109375	15
10	100	1	1	1	15 49222.46875	16
100	20	0	0	0	19 30735.168016825	17
100	20	0	1	1	6 33158.32421975	18
100	20	1	0	0	10 31703.666015625	19
100	20	1	1	1	7 35613.33203125	20
100	100	0	0	0	11 30408.84765625	21
100	100	0	1	1	13 35730.62109375	22
100	100	1	0	0	7 31885.578125	23
100	100	1	1	1	6 45209.17578125	24

FIGURE 34 – Résultat de l'entraînement

Se référer à l'annexe 1 pour les courbes d'apprentissage.

Analysons plus en détail l'influence des paramètres variables :

minMSE représente l'erreur en moyenne quadratique minimale obtenue sur le jeu de test.

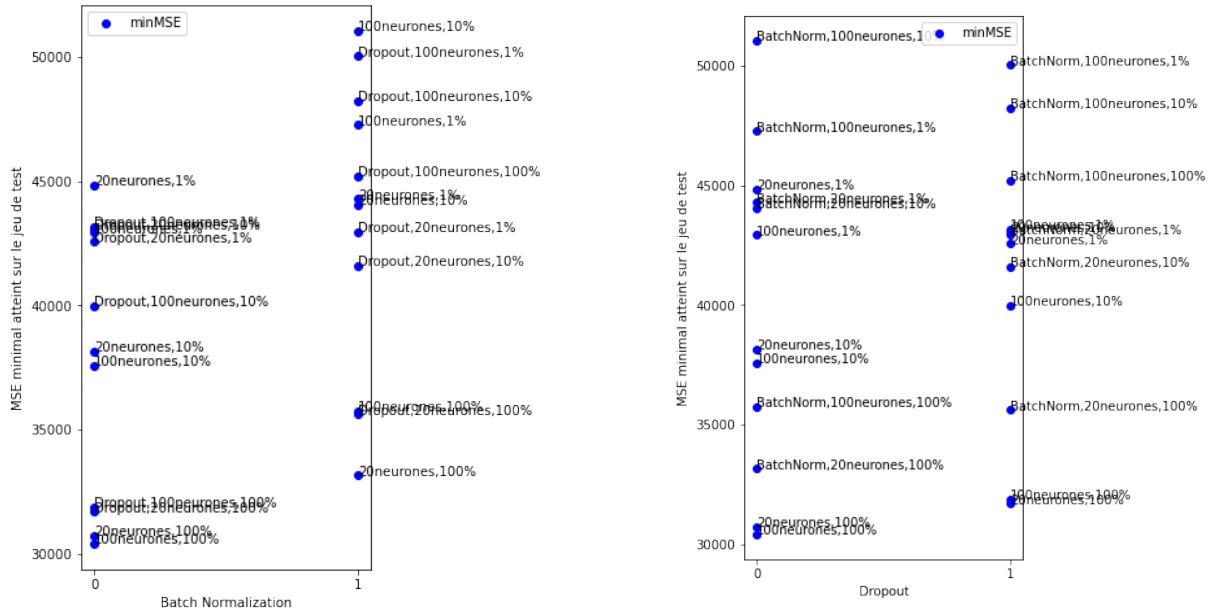


pourcentageDataset :

De manière prévisible, la taille du jeu de données influe positivement sur la précision du modèle. De plus 100 neurones offrent de meilleures performances que 20 neurones.

nombreDeNeurones :

les modèles les plus précis sont ceux sans Dropout, ni Batch Normalization. Un modèle avec Dropout est le deuxième plus précis. L'erreur quadratique moyenne n'étant pas le seul paramètre de fiabilité d'un modèle, on peut donc être amené à hésiter entre un modèle simple et un modèle avec Dropout. On retrouve également l'intérêt d'avoir un large jeu de données (100%). On observe un léger gain de performance avec un plus grand nombre de neurones. Cette remarque est analysée en détail plus loin dans le rapport.

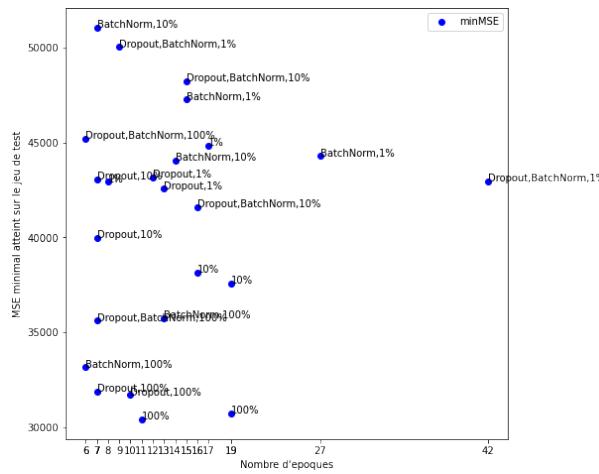


batchNormalization :

On obtient de meilleures performances sans Batch Normalization. L'erreur semble d'autant plus importante que le nombre de neurones augmente. Pour de la musique, essayer de normaliser les batch n'est peut être pas une bonne idée dans le sens où il n'y a pas de continuité entre les batch si les séquences ne sont pas sur la même gamme par exemple.

Dropout :

La perte de performance semble plus faible que pour le Batch Normalization mais reste tout de même présente. On remarque toutefois que pour 1% et 10% du jeu de données, le modèle avec Dropout offre de meilleures performances que le modèle sans. Cela semble pertinent puisque le Dropout sert à éviter le sur apprentissage et dégage une tendance générale. Cela est efficace sur un jeu de données restreint.



NombreEpoch :

l'erreur quadratique moyenne ne semble pas lié au nombre d'époque.

6.2.1 Analyse des courbes d'apprentissage (Annexe 1)

En représentant graphiquement l'erreur quadratique moyenne en fonction du nombre d'epoch selon différents paramètres (nombre de neurones, avec ou sans batch normalization, avec ou sans dropout), nous pouvons énoncer les conclusions suivantes : Lorsque le réseau est composé de 20 neurones, on observe

pour les jeux de données réduits un phénomène de surapprentissage (la courbe de test est éloignée de la courbe d'apprentissage), avec drop out ce problème est évité puisque les courbes sont plus proches. En revanche il n'est pas possible de tirer de conclusion sur l'utilité de la batch normalization : en effet même si les courbes sont plus proche d'une affine, l'erreur augmente sensiblement et un pic d'erreur apparaît. Ce pic est présent pour l'étude de tous les modèles avec batch normalization. Ce pic peut être expliqué par de mauvais paramètres d'activation puisque l'initialisation de ceux-ci est arbitraire. Indépendamment du pic d'erreur, puisque l'erreur ne diminue pas avec batch normalization, on considérera que cela n'est pas utile. Sur le jeu de données complet, on observe principalement que l'utilisation de drop out diminue le nombre d'epoch mais l'erreur est sensiblement la même (un peu plus basse sans utiliser de drop out) et de même l'utilisation combinée de batch normalisation et de drop out montre un modèle performant (erreur proche du modèle de base) et des courbes d'apprentissages assez proches (témoignant de la qualité de l'apprentissage) avec un nombre d'epoch plus bas. En résumé avec 20 neurones, l'erreur est la plus basse sans utiliser de drop out ni de batch normalization. Cependant l'utilisation des deux paramètres supplémentaires permet d'améliorer la qualité de l'apprentissage ainsi que de diminuer le nombre d'epoch sans augmenter l'erreur considérablement.

On remarque qu'en utilisant 100 neurones, le drop out est définitivement impactant sur la vitesse de convergence des paramètres de notre RNN, la batch normalization n'est cependant pas utile dans ce contexte ci.

En conclusion, le meilleur modèle retenue pour le jeu de données complet serait celui à 100 neurones sans Drop out ni batch normalization, en effet c'est celui qui donne l'erreur la plus basse. On retiendra quand même que l'erreur diminue à mesure que le nombre de données considérées augmente et que le drop out permet de réduire considérablement le temps de convergence sans pour autant perdre (massivement) en qualité.

6.3 Amélioration

D'après l'analyse précédente, pour améliorer notre modèle, on peut augmenter la taille du jeu de données (impossible dans notre cas) ou augmenter le nombre de neurones sur notre modèle le plus performant. Étudions donc l'influence du nombre de neurones sur 100% du jeu de données, sans Dropout ni Batch Normalization :

Commençons d'abord par les courbes d'apprentissage :

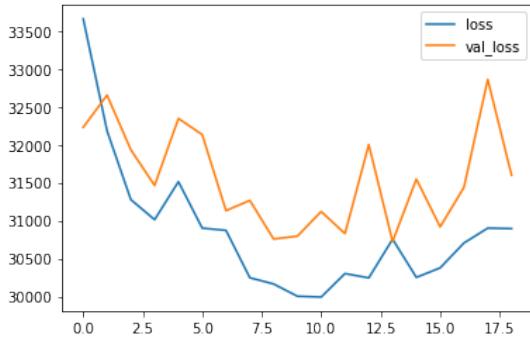


FIGURE 35 – 20 neurones

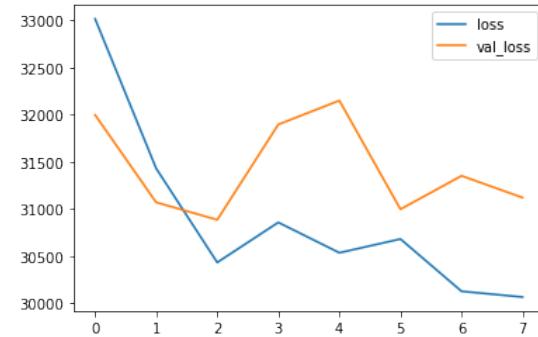


FIGURE 36 – 60 neurones

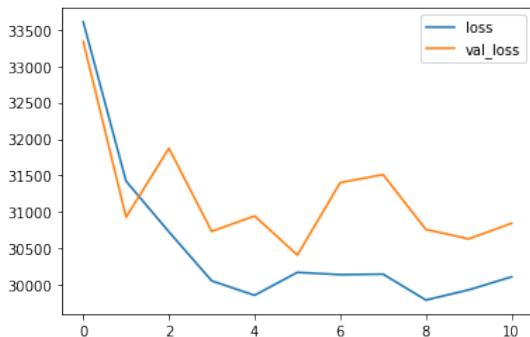


FIGURE 37 – 100 neurones

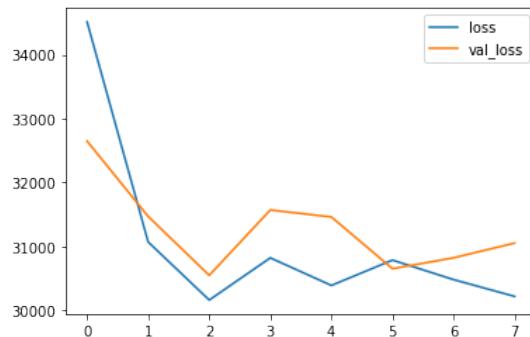


FIGURE 38 – 140 neurones

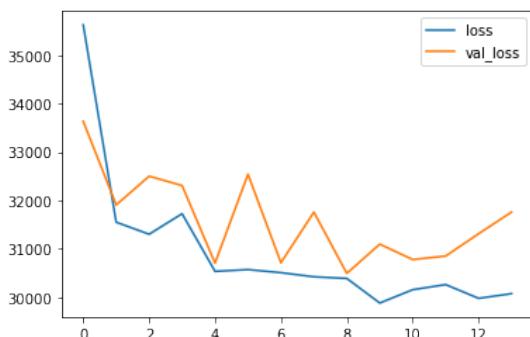


FIGURE 39 – 180 neurones

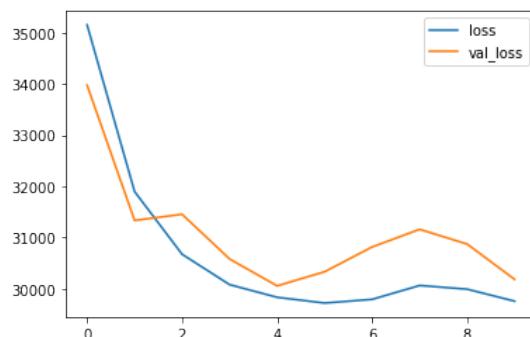


FIGURE 40 – 220 neurones

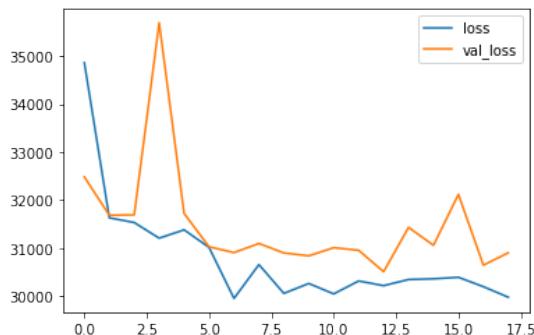


FIGURE 41 – 260 neurones

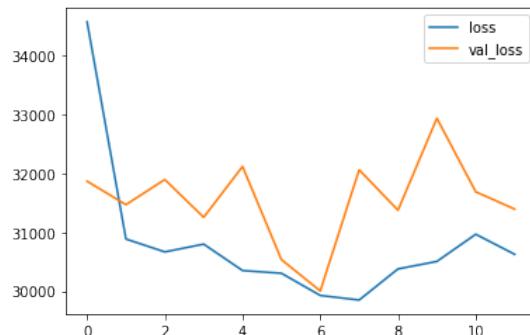


FIGURE 42 – 300 neurones

Puis l'erreur quadratique moyenne minimale atteinte en fonction du nombre de neurones :

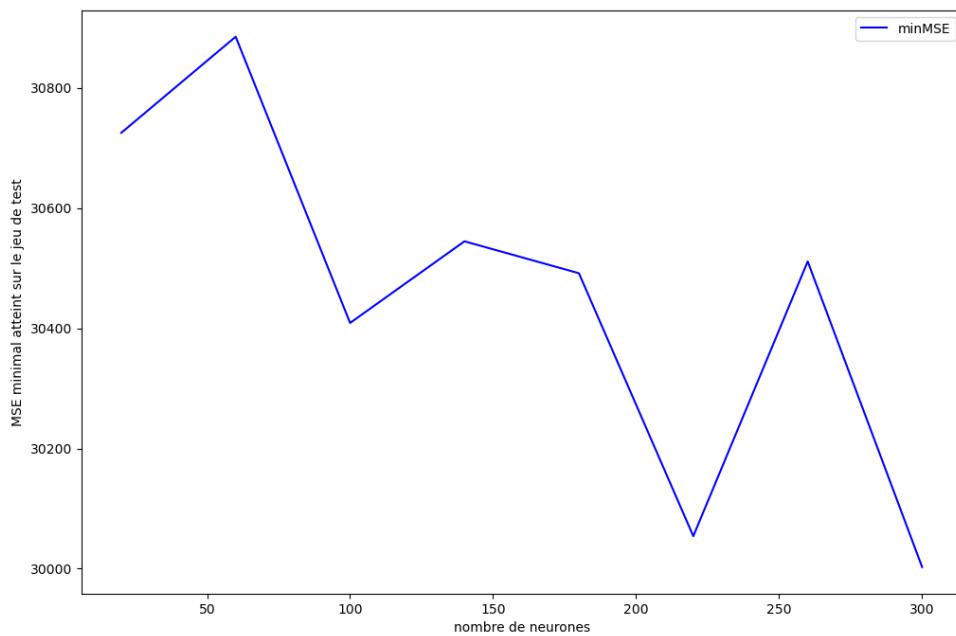


FIGURE 43 – Résultat de l'entraînement

L'erreur semble diminuer sans toute fois être continue. Il va falloir faire un compromis entre le nombre de neurones et la durée d'entraînement disponible.

Un modèle avec 300 neurones offre les meilleurs résultats mais le modèle avec 220 neurones a la courbe d'apprentissage la plus continue. Comme on compare les performances de deux modèles sur le même jeu de test, on ne peut prédire les performances sur un nouveau jeu de test. Toutefois, avec un modèle qui a un apprentissage qui performe de manière continue, on peut s'attendre à avoir des performances correctes sur un nouveau jeu de test. On aurait ainsi pu introduire un jeu de validation en plus mais perdre des données d'entraînement, ou faire confiance à la continuité de l'apprentissage.

Il est donc cohérent de générer de la musique à l'aide du modèle de 220 neurones, sans Dropout, ni normalisation de Batch et avec 100% du jeu de données (70% d'entraînement et 30% de test).

7 Génération de musique

7.1 Données en jeu

Pour pouvoir analyser et comprendre les morceaux musicaux créés par notre réseau de neurones, nous avons décidé de lui fournir N notes existantes d'un morceau afin qu'il prédise la 16 ème (on réitère le processus en décalant les notes en entrée pour obtenir au final un morceau complet). Ainsi il sera possible de comparer le morceau original du morceau reproduit par notre modèle. Le choix de fournir 15 notes est purement arbitraire, il est intuitif de penser que plus on fournit de notes initialement, plus le modèle sera précis et juste sur sa prédition. Ainsi Nous avons fourni les 15 premières notes de 4 morceaux différents de notre dataset (sélectionnés aléatoirement). On présentera dans un premier temps une analyse des valeurs de notes créées pour chaque morceau, puis une analyse de la durée des notes, et enfin de la durée entre les notes.

7.2 Analyse

Nous allons maintenant analyser les output de notre RNN, pour ce faire nous étudions statistiquement les fichiers résultats associés aux morceaux créés, pour les courbes d'erreur, on affichera seulement la courbe d'un morceau sur les 4 générés car elles sont toutes similaires. On affichera les profils des morceaux jusqu'à la 200ème note seulement, cela suffit pour analyser l'entiereté du résultat tout en améliorant la lisibilité.

7.2.1 Valeur des notes

Les courbes suivantes représentent nos morceaux de musiques, en orange le morceau conçu par notre RNN, en bleu le morceau original :

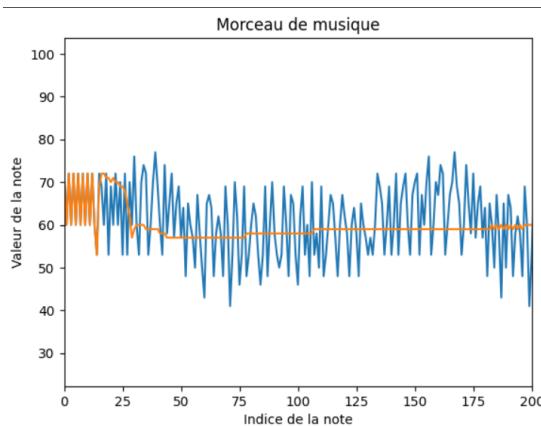


FIGURE 44 – Valeur des notes du morceau 1

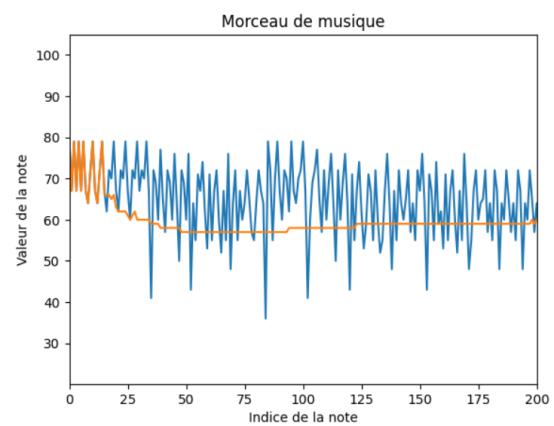


FIGURE 45 – Valeur des notes du morceau 2

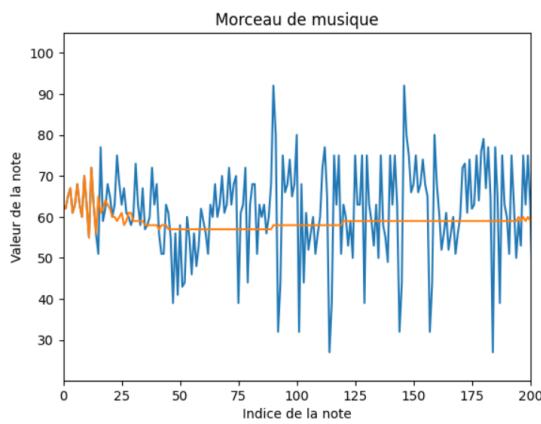


FIGURE 46 – Valeur des notes du morceau 3

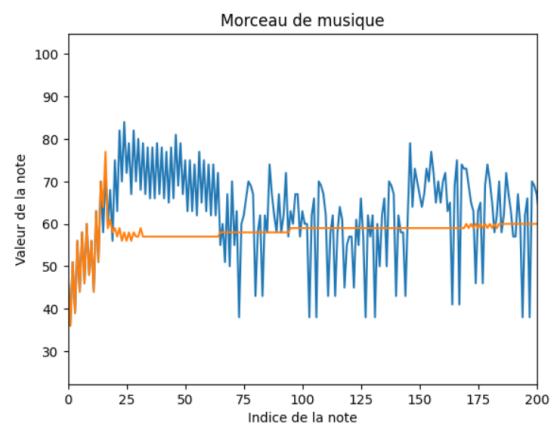


FIGURE 47 – Valeur des notes du morceau 4

On observe dans un premier temps un phénomène de convergence et de répétition d'une note pour chaque morceau, la note 60. Ce phénomène était attendu et peut s'expliquer par le fait que notre réseau n'est peut être pas optimal : Les paramètres de ce dernier peuvent être améliorés, nous n'utilisons ni batch normalization, ni drop out qui sont des méthodes classiques permettant d'améliorer la qualité de l'entraînement (notre modèle semblait plus performant sans), le sur apprentissage n'est pas parfaitement contrôlé. Ce sont des problèmes que nous aurions pu résoudre, mais par soucis de temps nous préférerons nous concentrer sur l'étude et la compréhension de nos résultats en l'état. De plus, même si la gamme des morceaux semble correcte à chaque fois, les valeurs de notes s'éloignent assez vite des valeurs correctes et notre modèle ne retranscrit pas la versatilité de gamme des morceaux, il est clair que les notes créées restent autour de la note moyenne de chaque morceau. On peut représenter l'erreur cumulée des valeurs des notes sur chaque morceau ainsi que l'erreur quadratique par note pour chaque morceau :

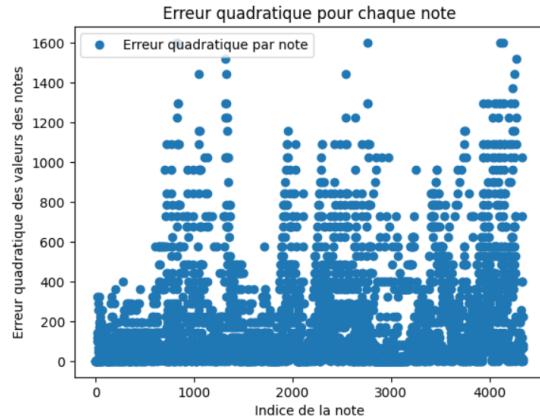


FIGURE 48 – EQM des valeurs de note du morceau 1

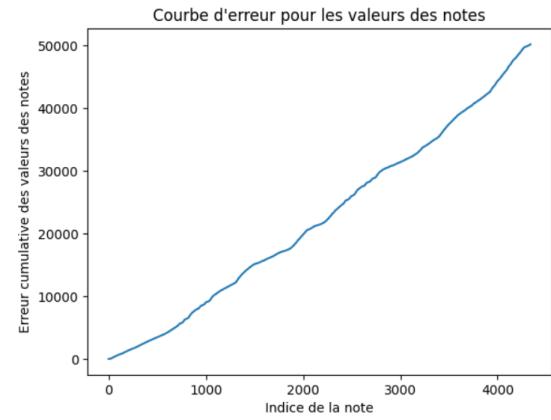


FIGURE 49 – erreur cumulée des valeurs de note du morceau 1

On observe comme attendu une erreur cumulée qui augmente à mesure qu'on avance dans le morceau, cela s'explique par le fait que notre modèle présente une erreur non négligeable. De ce fait, certaines prédictions sont mauvaises comme on peut le voir sur les graphiques présentant les moyennes quadratiques par note, cependant on note tout de même que beaucoup de notes sont prédites (presque) correctement puisque le nuage de points est assez concentré autour de l'axe 0. En résumé, notre modèle n'a pas été capable de prédire parfaitement les suites de notes de morceaux tests, cela s'explique par le fait que l'entraînement a donné un modèle dont l'erreur était assez éloignée de 0. Néanmoins il est important de noter que les morceaux sont joués dans la bonne gamme, autour de la note moyenne déterminée lors de l'analyse du jeu de données.

7.3 Durée des notes

Les courbes suivantes représentent les durées des notes de chaque morceau :

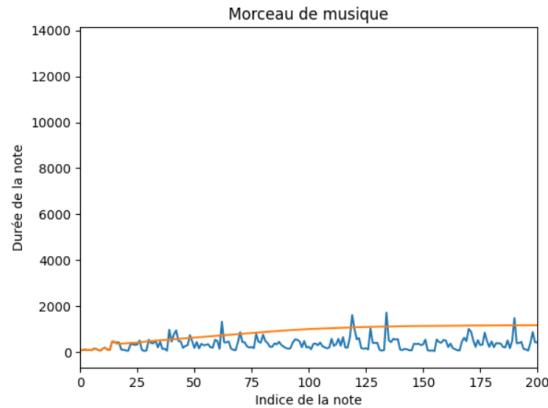


FIGURE 50 – Durée des notes du morceau 1

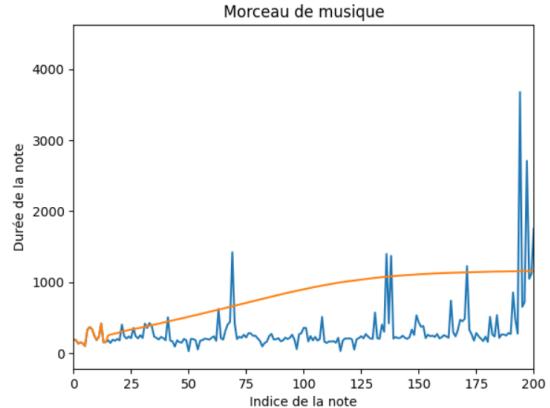


FIGURE 51 – Durée des notes du morceau 2

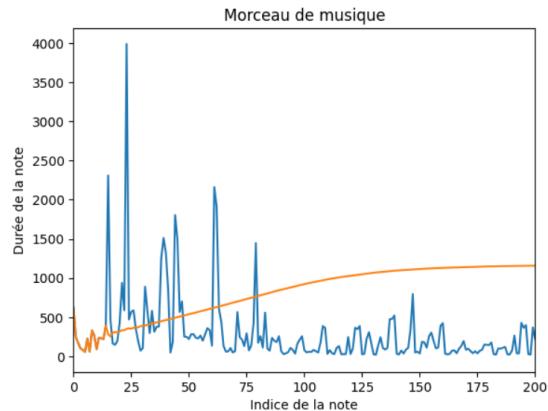


FIGURE 52 – Durée des notes du morceau 3

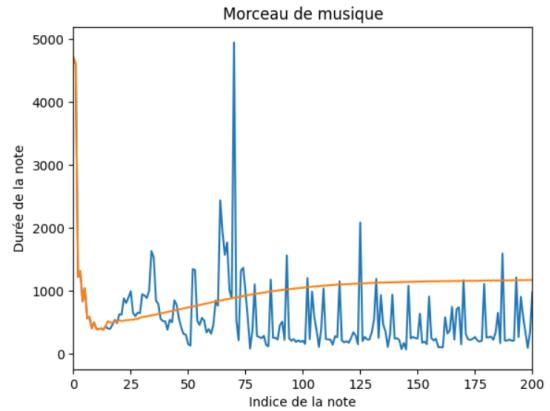


FIGURE 53 – Durée des notes du morceau 4

On observe un phénomène de répétition conséquent : la courbe donnant la durée des notes du morceau artificiel augmente continuellement jusqu'à être asymptote de la droite constante $y=1179$ qui correspond à la médiane des durées sur les morceaux mis en comparaison. Cela montre que notre modèle ne peut pas suivre les divergences fortes des durées des notes dans un morceau réel, il se contente de stabiliser la durée des notes à la prétendue durée moyenne des notes du morceau. Le problème ici est donc différent de celui rencontré pour les valeurs de notes, en effet le problème réside surtout dans le manque de corrélation entre les durées des notes. De ce fait, le modèle considère ce paramètre comme aléatoire et n'est pas capable de produire des durées cohérentes. On représente ici l'erreur cumulée et l'erreur quadratique sur chaque durée de note pour les différents morceaux :

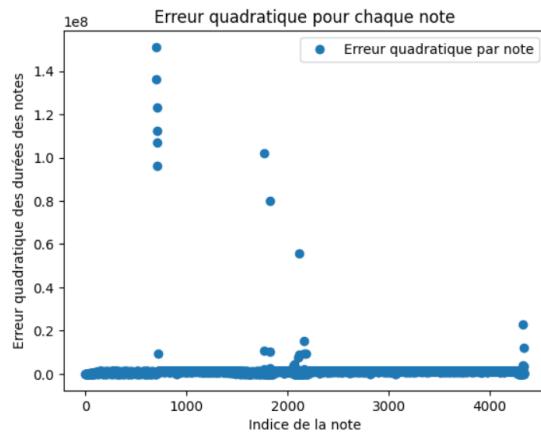


FIGURE 54 – EQM sur la durée des notes du morceau 1

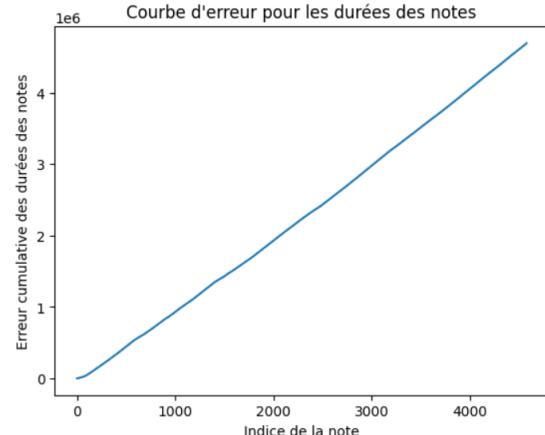


FIGURE 55 – Erreur cumulée sur la durée des notes du morceau 1

On observe une erreur cumulée croissante comme prévu, avec des valeurs extrême (car les différences pour les durées des notes sont très grandes une fois mise au carré) qui faussent légèrement l'échelle, en prenant seulement les notes en dessous la plus basse des valeurs d'erreur extrême on obtient les graphiques suivants :

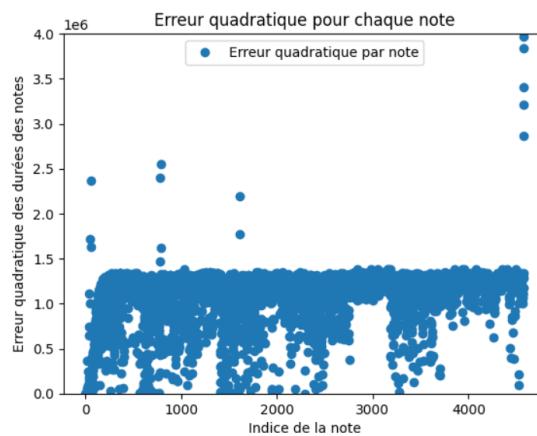


FIGURE 56 – EQM sur la durée des notes du morceau 1 (sans les pics d'erreur)

Il est clair que le nuage est assez proche de l'axe 0 (malgré une erreur non négligeable) puisque les notes retenues sont les notes proches de la moyenne. Notre modèle est donc bien capable de s'approcher des durées de note correcte sans pour autant les reproduire parfaitement.

7.4 Durée entre les notes

Les courbes suivantes présentent les durées avant la note suivante pour chaque note :

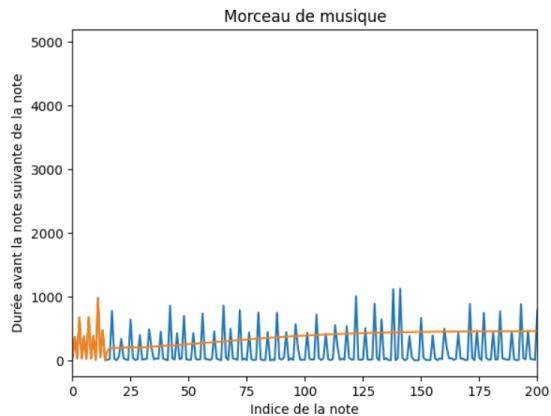


FIGURE 57 – Durée entre les notes du morceau 1

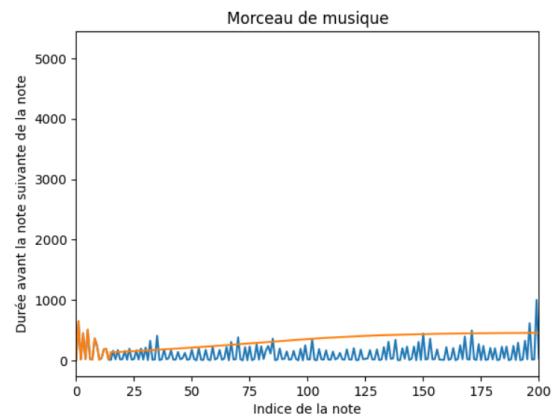


FIGURE 58 – Durée entre les notes du morceau 2

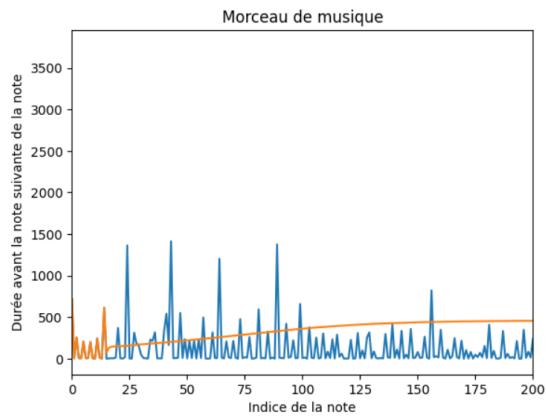


FIGURE 59 – Durée entre les notes du morceau 3

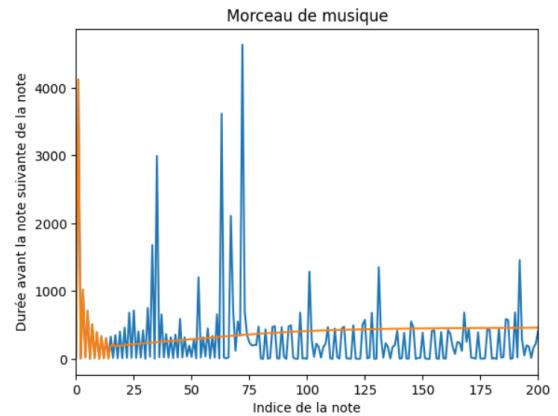


FIGURE 60 – Durée entre les notes du morceau 4

On observe le même phénomène que pour les durées de note, ce facteur la est aussi considéré aléatoire par le modèle et ce dernier ajuste simplement le paramètre à la moyenne des durées entre les notes pour les morceaux pris en input. Représentons enfin l'erreur quadratique associée à chaque note ainsi que l'erreur cumulée :

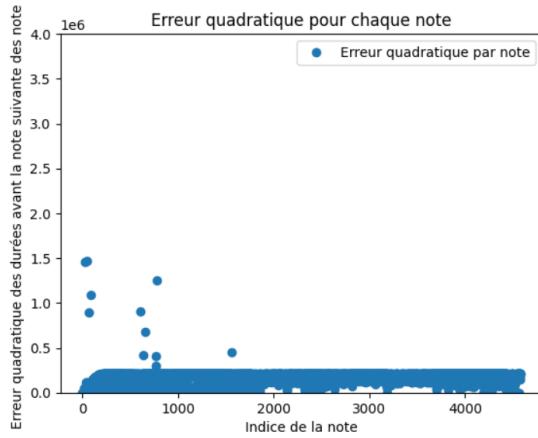


FIGURE 61 – EQM sur la durée entre les notes du morceau 1

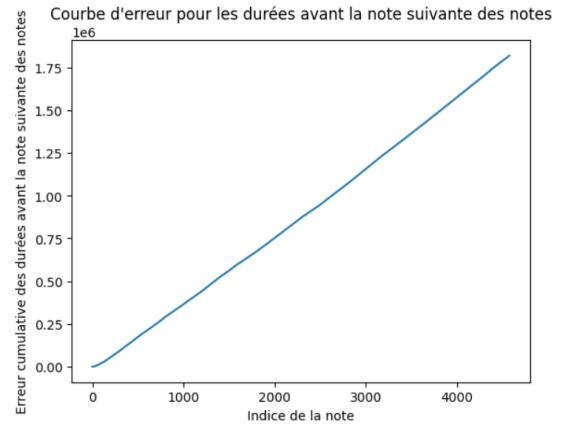


FIGURE 62 – Erreur cumulée sur la durée entre les notes du morceau 1

Les graphiques sont très semblables aux graphiques sur la durée des notes, on peut donc tirer les mêmes conclusions. C'est logique puisque chacun des deux paramètres dépend d'un facteur temps qui est assez arbitraire et dépend de la créativité du compositeur. Alors notre modèle n'est pas capable de se montrer "créatif".

7.5 Conclusion sur les résultats

En conclusion, l'analyse des résultats de notre modèle de génération musicale révèle des tendances de convergence autour d'une seule note, une difficulté à capturer la diversité des valeurs de notes, et un problème notable dans la génération des durées de notes ainsi que des durées entre les notes. Les erreurs cumulées augmentent progressivement, suggérant une perte de précision au fil du morceau. Les pistes d'amélioration comprennent l'ajustement des paramètres du modèle, l'exploration de l'ajout de techniques comme la batch normalization et le dropout, et une meilleure prise en compte des relations temporelles complexes entre les notes. Malgré les problèmes énoncés, le modèle parvient à maintenir les morceaux dans la bonne gamme, soulignant la nécessité d'équilibrer la créativité du modèle avec une précision accrue dans la génération musicale.

Références

- [Cho23] François Chollet. Documentation keras, 2023.
- [Col] Collections. collections, types de données.
- [DAT] DATATab. test de normalité.
- [HSR⁺18] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. The maestro dataset, 2018.
- [Joh15] Daniel Johnson. Composing music with recurrent neural networks, 2015.
- [Lin18] Joe Linux. Introduction à la programmation midi avec python, mido et rtmidi, 2018.
- [Liu23] Yating Liu. Introduction à l'apprentissage statistique. *Université Paris-Dauphine*, 2023.
- [Num06] Numpy. Numpy documentation, 2006.
- [Sci] Scipy. Scipystat documentation.
- [Ten15] TensorFlow. Documentation tensorflow pour python, 2015.
- [Ten21] Tensorflow. Générer de la musique avec un rnn, 2021.

8 Annexe 1 : Courbes d'apprentissage

Pourcentage du dataset	nombre de neurones	Dropout	batch Normalization	epoch	minMSE	Numéro de photo
1	20	0	0	1	17 44827.609375	1
1	20	0	1	1	27 44290.59765625	2
1	20	1	0	0	13 42573.31640625	3
1	20	1	1	1	42 42944.87109375	4
1	100	0	0	0	8 42927.71875	5
1	100	0	1	1	15 47274.40625	6
1	100	1	0	0	12 43183.86328125	7
1	100	1	1	1	9 50033.05079125	8
10	20	0	0	0	16 38913.5703125	9
10	20	0	1	1	14 44072.8046875	10
10	20	1	0	0	7 43057.47265625	11
10	100	0	0	0	16 41597.72265625	12
10	100	0	1	1	19 37595.12109375	13
10	100	1	0	0	7 51067.93359375	14
10	100	1	1	0	7 39991.2109375	15
10	100	1	1	1	15 48222.46875	16
100	20	0	0	0	19 30725.166015625	17
100	20	0	1	1	6 33158.32421875	18
100	20	1	0	0	10 31703.666015625	19
100	20	1	1	1	7 35613.33203125	20
100	100	0	0	0	11 30408.84765625	21
100	100	0	1	1	13 35730.62109375	22
100	100	1	0	0	7 31885.578125	23
100	100	1	1	1	6 45209.17578125	24

FIGURE 63 – Résultat de l'entraînement

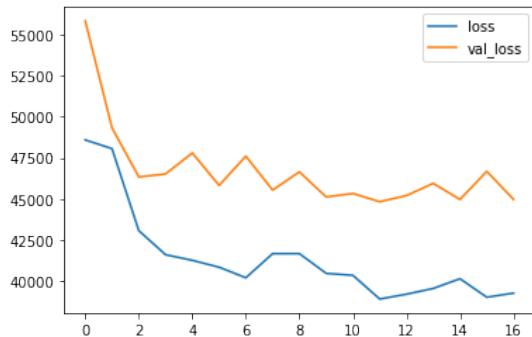


FIGURE 64 – 1

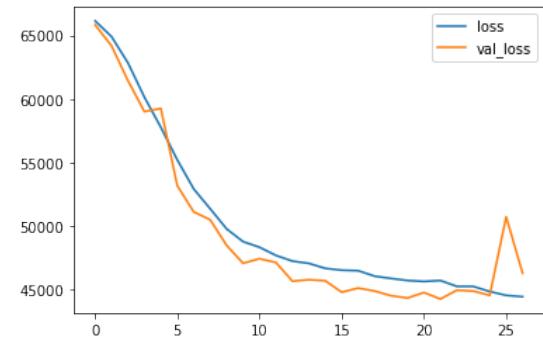


FIGURE 65 – 2

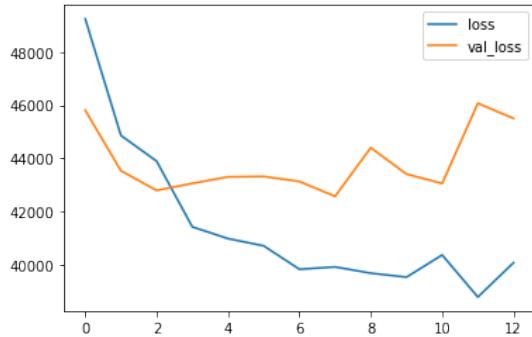


FIGURE 66 – 3

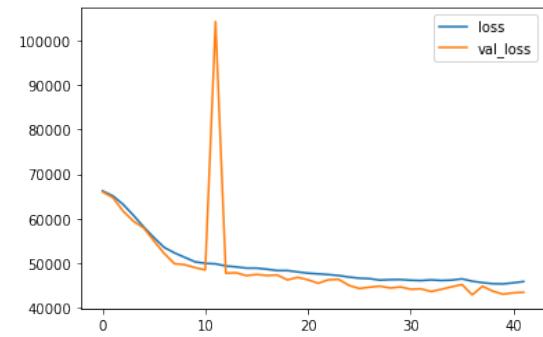


FIGURE 67 – 4

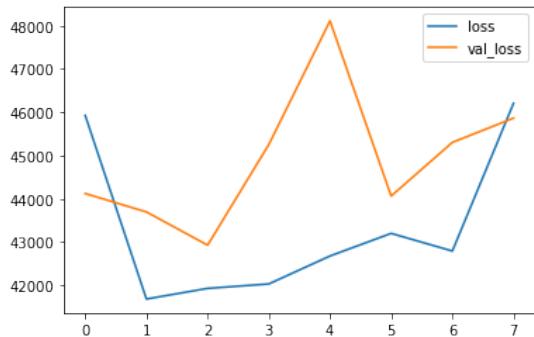


FIGURE 68 – 5

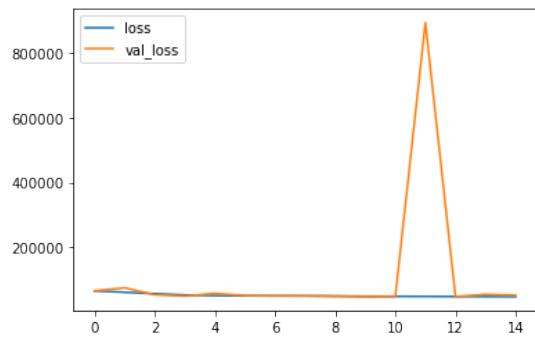


FIGURE 69 – 6

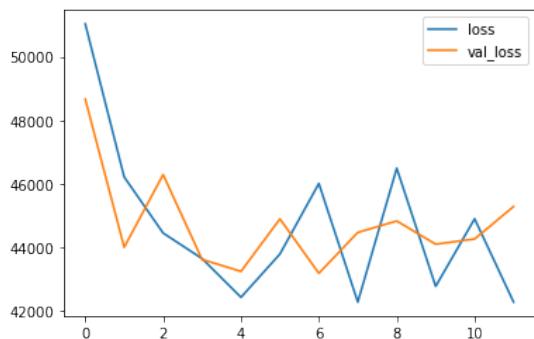


FIGURE 70 – 7

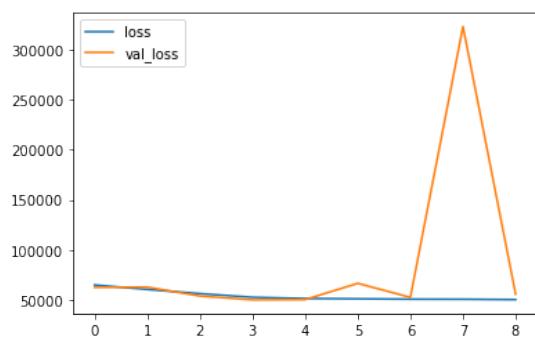


FIGURE 71 – 8

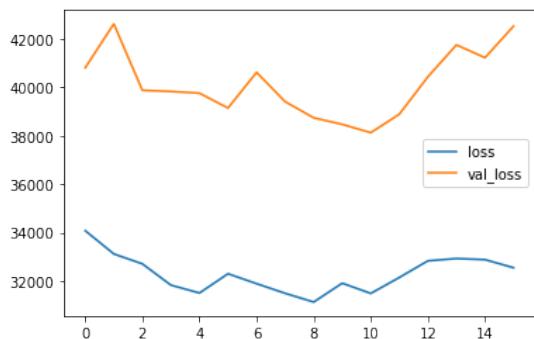


FIGURE 72 – 9

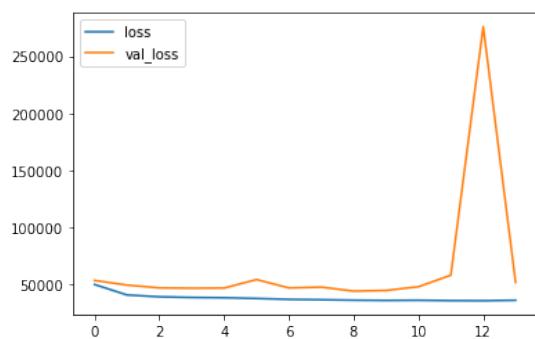


FIGURE 73 – 10

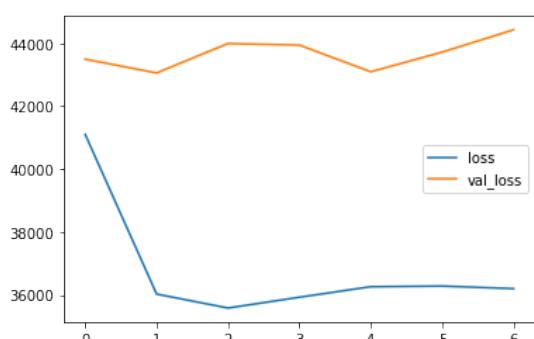


FIGURE 74 – 11

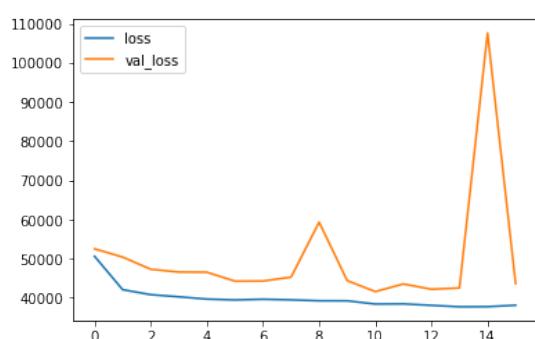


FIGURE 75 – 12

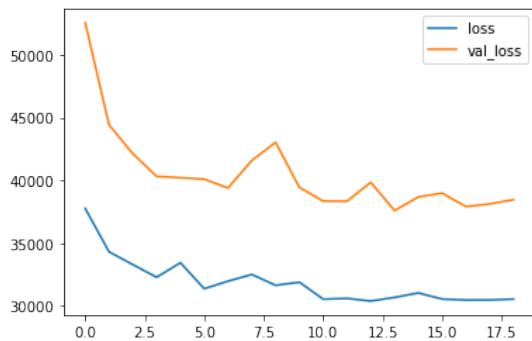


FIGURE 76 - 13

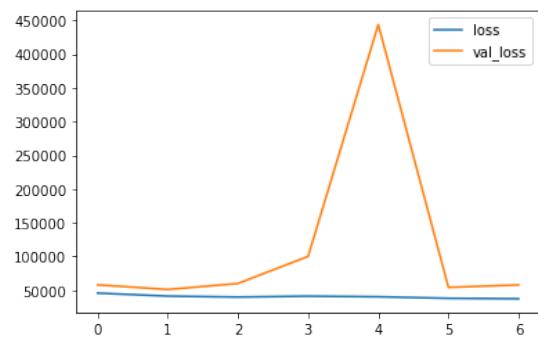


FIGURE 77 - 14

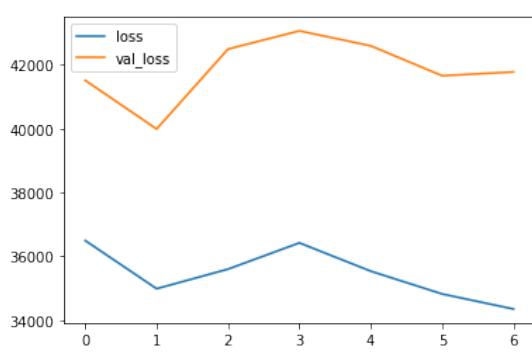


FIGURE 78 - 15

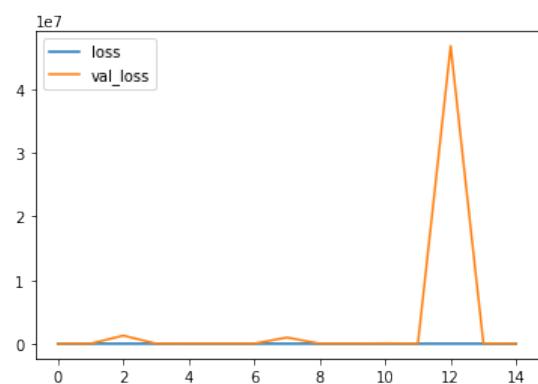


FIGURE 79 - 16

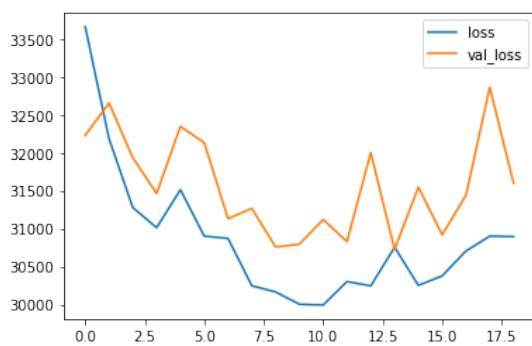


FIGURE 80 - 17

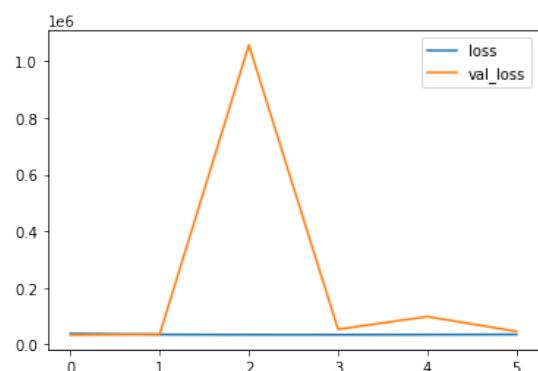


FIGURE 81 - 18

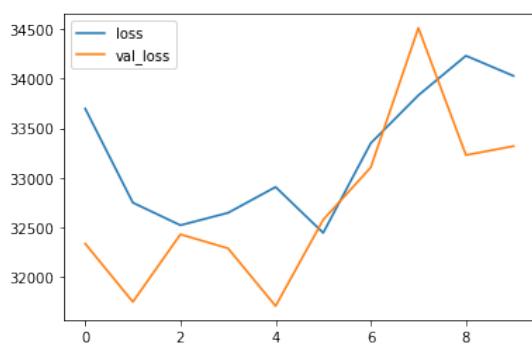


FIGURE 82 - 19

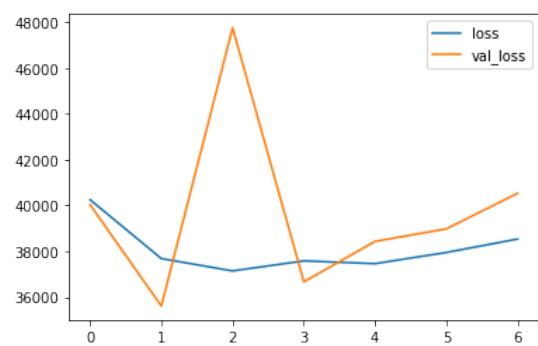


FIGURE 83 - 20

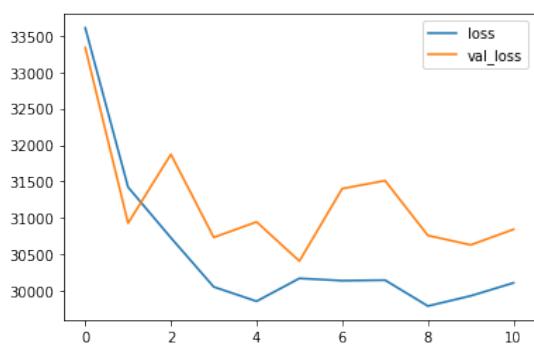


FIGURE 84 – 21

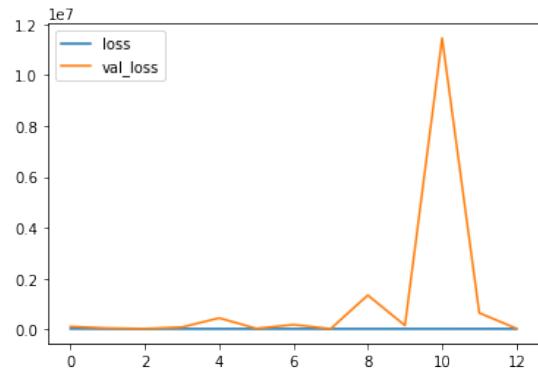


FIGURE 85 – 22

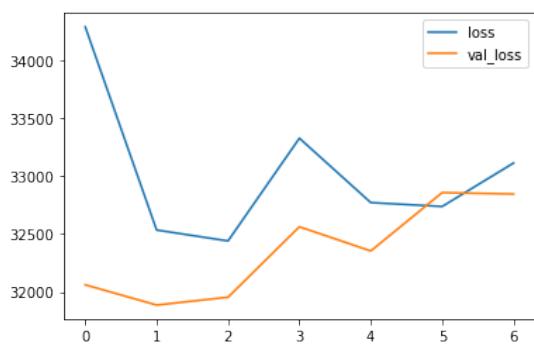


FIGURE 86 – 23

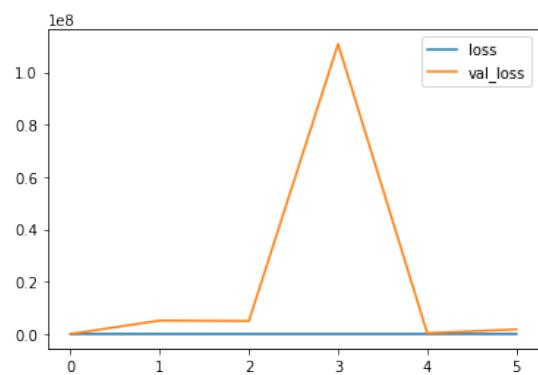


FIGURE 87 – 24