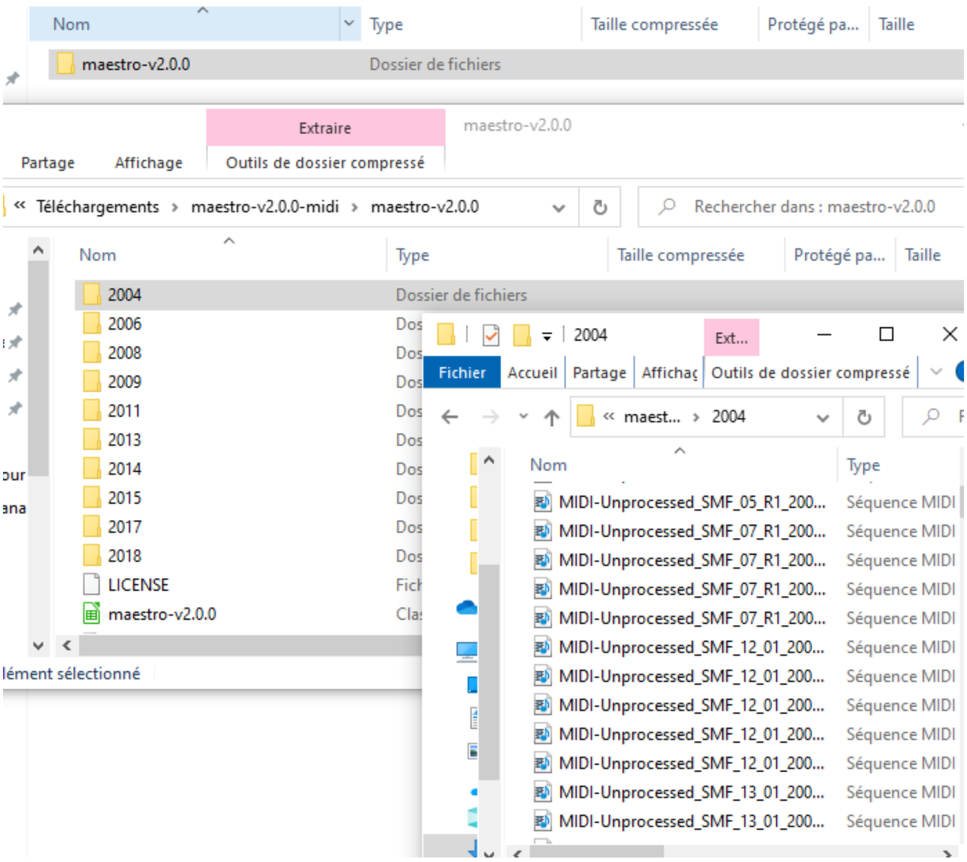
Homework : Music generation by lstm

Student : Ait mansour hamza

Dataset


Extraire
maestro-v2.0.0
Partage
Affichage
Outils de dossier compressé
« Téléchargements › maestro-v2.0.0-midi › maestro-v2.0.0
Rechercher dans : maestro-v2.0.0
lément sélectionné
2004
Ext...
Fichier
Accueil
Partage
Affichag
Outils de dossier compressé
« maest... › 2004

Nom	Type	Taille compressée	Protégé pa...	Taille
maestro-v2.0.0	Dossier de fichiers			

Nom	Type	Taille compressée	Protégé pa...	Taille
2004	Dossier de fichiers			
2006	Dos			
2008	Dos			
2009	Dos			
2011	Dos			
2013	Dos			
2014	Dos			
2015	Dos			
2017	Dos			
2018	Dos			
LICENSE	Fich			
maestro-v2.0.0	Cla			

Nom	Type
MIDI-Unprocessed_SMF_05_R1_200...	Séquence MIDI
MIDI-Unprocessed_SMF_07_R1_200...	Séquence MIDI
MIDI-Unprocessed_SMF_07_R1_200...	Séquence MIDI
MIDI-Unprocessed_SMF_07_R1_200...	Séquence MIDI
MIDI-Unprocessed_SMF_07_R1_200...	Séquence MIDI
MIDI-Unprocessed_SMF_12_01_200...	Séquence MIDI
MIDI-Unprocessed_SMF_12_01_200...	Séquence MIDI
MIDI-Unprocessed_SMF_12_01_200...	Séquence MIDI
MIDI-Unprocessed_SMF_12_01_200...	Séquence MIDI
MIDI-Unprocessed_SMF_12_01_200...	Séquence MIDI
MIDI-Unprocessed_SMF_13_01_200...	Séquence MIDI
MIDI-Unprocessed_SMF_13_01_200...	Séquence MIDI

Number of files : 1282

Instrument : Piano

Training : started with 5 files and experimented in the last version with 10 (10 good enough so far)

Testing : the whole set of data (1282 files)

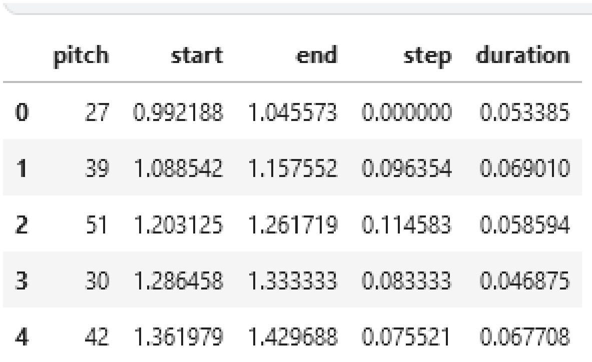
Number of notes in those 10 files combined : 45847

Extract notes from MIDI file

midi_to_notes function + création des variables pour chaque note

- **pitch** (equality of the sound as a MIDI note number 1..60);
- **step** (time elapsed from the previous note);
- **duration** (how long the note will be playing in seconds - the difference between the note end and note start times).
- **Start**

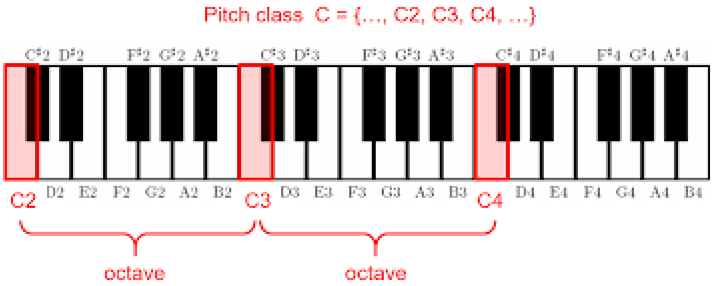
- **End**



	pitch	start	end	step	duration
0	27	0.992188	1.045573	0.000000	0.053385
1	39	1.088542	1.157552	0.096354	0.069010
2	51	1.203125	1.261719	0.114583	0.058594
3	30	1.286458	1.333333	0.083333	0.046875
4	42	1.361979	1.429688	0.075521	0.067708

To make things easier we are gonna interpret the note by their names rather than their pitch, so we are gonna use the function note_number_to_name of prettyMidi to convert from the numeric pitch values to note names. The note name shows the type of note, accidental and octave number

C#2 D#4 A2 and so on...



Training phase

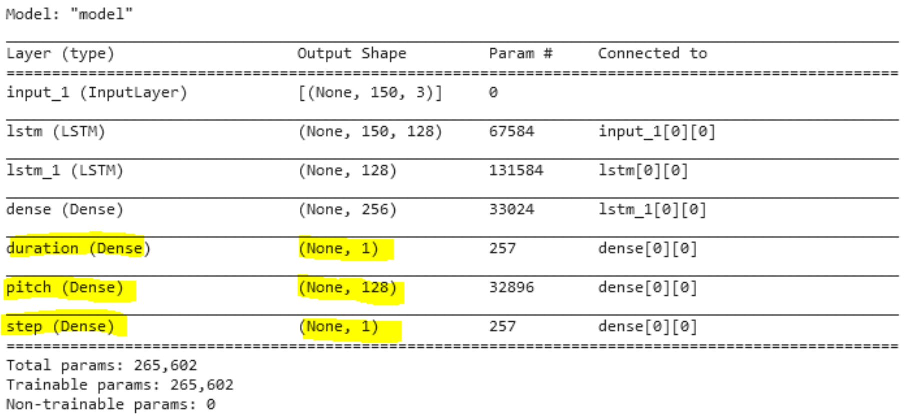
We need to train the model on group of sequences of notes.

Sequence shape (150,3). 150 as size of seq and 3 as number of classes : pitch, step , duration.

Each example will consist of a sequence of notes as the input features for the net, and next note as the label. In this way, the model will be trained to predict the next note in a sequence.

Model

Expected output in yellow, the model should give as an output for every note duration pitch step



```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 150, 3)]     0

 lstm (LSTM)                    (None, 150, 128)     67584       input_1[0][0]

 lstm_1 (LSTM)                  (None, 128)          131584      lstm[0][0]

 dense (Dense)                  (None, 256)          33024       lstm_1[0][0]

 duration (Dense)               (None, 1)            257         dense[0][0]

 pitch (Dense)                  (None, 128)          32896       dense[0][0]

 step (Dense)                   (None, 1)            257         dense[0][0]

==================================================================================================
Total params: 265,602
Trainable params: 265,602
Non-trainable params: 0
```

Activation : Relu

Optimizer : Adam | Learning rate : 0.005

Losses :

```
{'loss': 0.15689390897750854, // individually
 'duration_loss': 0.13079428672790527,
 'pitch_loss': 4.855881214141846,
 'step_loss': 0.03361739218235016}
```

Music Generation

Using the model we r going to generate notes.

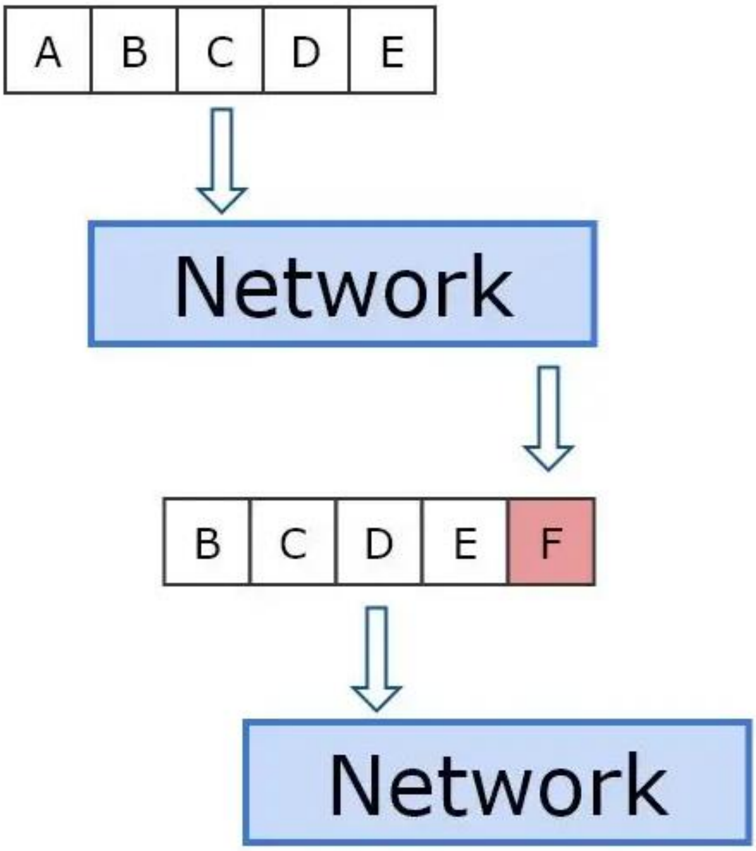
prerequisite:

- A starting sequence of notes. (Using Testing dataset 4914 notes to create sequences)

- A function that predicts next note from sequence of notes (using model.predict)

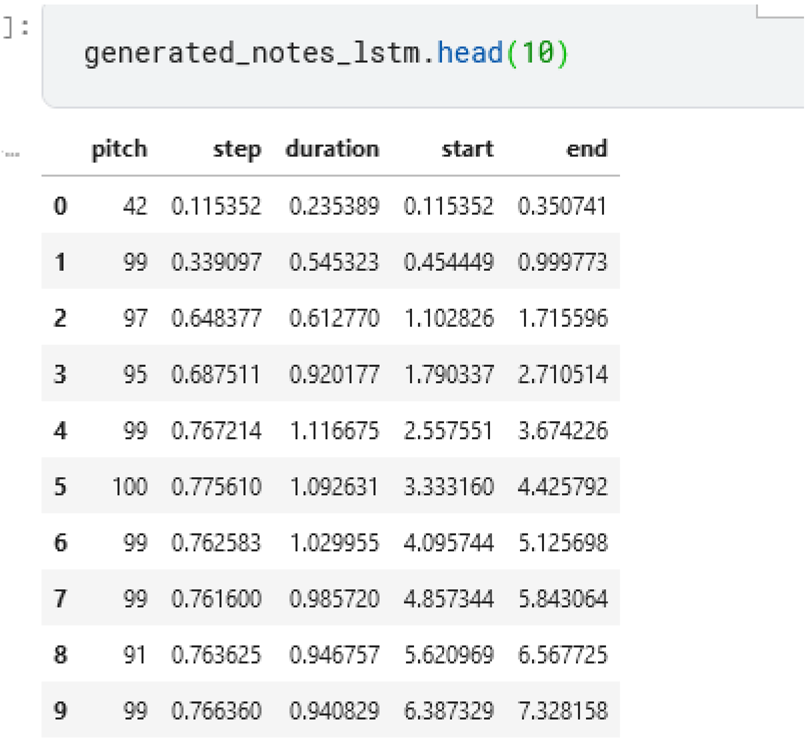
The input sequence should be updated as below :

F is the note generated by sequence ABCDE, the next iteration we pass the sequence BCDEF as input and so on.



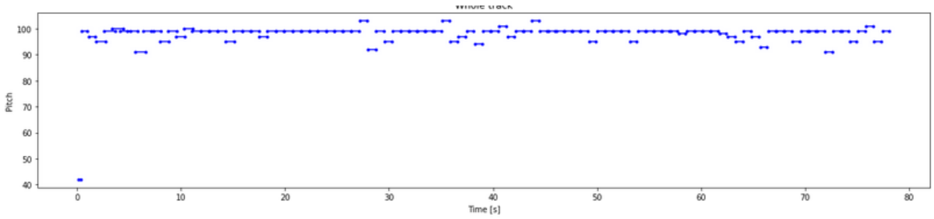
Results : (check output.mid)

A 30 secondes songs : 62 notes generated (I limited the number of predictions to 100)



```
generated_notes_lstm.head(10)
```

	pitch	step	duration	start	end
0	42	0.115352	0.235389	0.115352	0.350741
1	99	0.339097	0.545323	0.454449	0.999773
2	97	0.648377	0.612770	1.102826	1.715596
3	95	0.687511	0.920177	1.790337	2.710514
4	99	0.767214	1.116675	2.557551	3.674226
5	100	0.775610	1.092631	3.333160	4.425792
6	99	0.762583	1.029955	4.095744	5.125698
7	99	0.761600	0.985720	4.857344	5.843064
8	91	0.763625	0.946757	5.620969	6.567725
9	99	0.766360	0.940829	6.387329	7.328158



You can find the source code and the output music :

[Link to source code]()