Music Generation using LSTM

Fabrizio Rocco - Davide Rosatelli - Roberto Colangelo

Luiss Guido Carli

Index

Our Repository

- Before starting, you can find our full project on our GitHub repo:

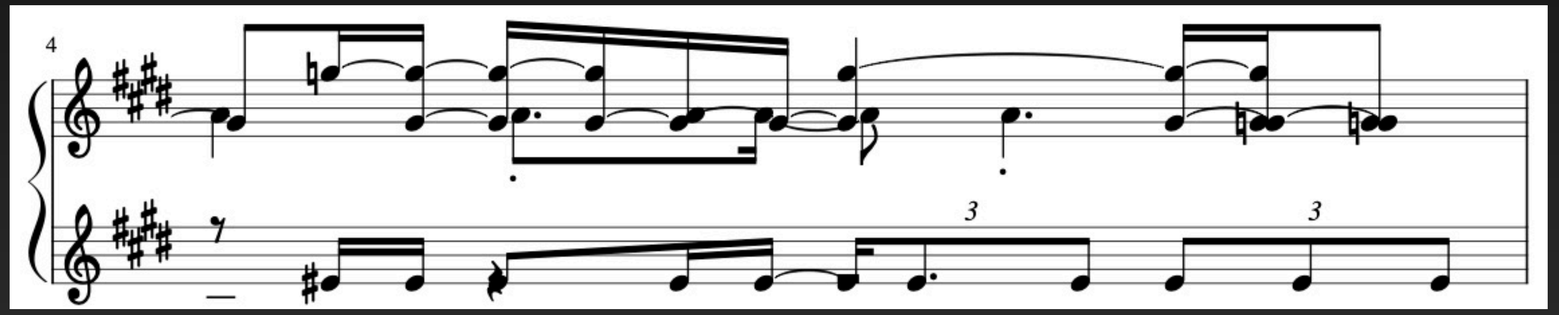
https://github.com/fabriziorocco/MusicGeneration

The problem

- Music has always been one of the most characterizing aspects of the mankind, that, over the time, has gathered people in moments of joy and lightheartedness.
- We have decided to entrust the task of **making a machine able to generate music.**
- **Why LSTM?**

Goal

- Our goal is to make an algorithm able to predict **the next sequence** of notes and chords.



After ?

Sequence of fixed length

Methodology

- 2016 - WaveNet -> deepmind.com
- 2017 - Magenta -> magenta.tensorflow.org

- Since 2016 Google AI department explores and dominates this field. Their models are the actual State of the Art.

- The literature available on this theme proposes **3 different possible architectures**:

GAN - **LSTM (Recurrent Networks)** - 1D Convolutional

The two main problems we have dealt with

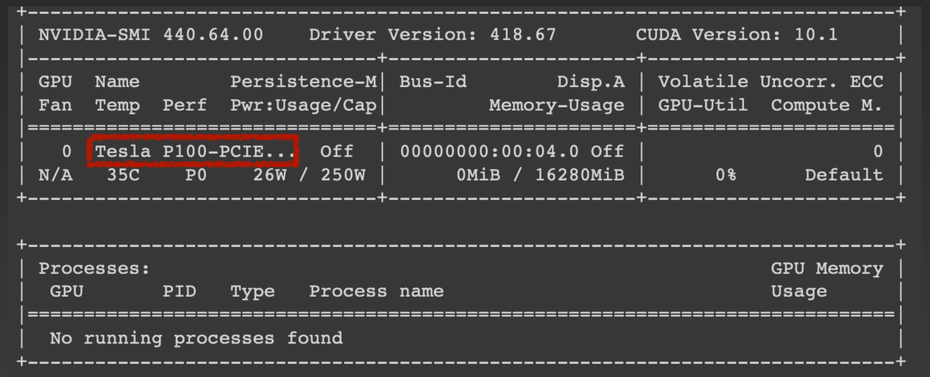
Data Format and Interaction

- User Friendly or Computer Friendly ?
- The choice of MIDI files

Computational Power

- Huge Training times
- Poor computational power and GPUs

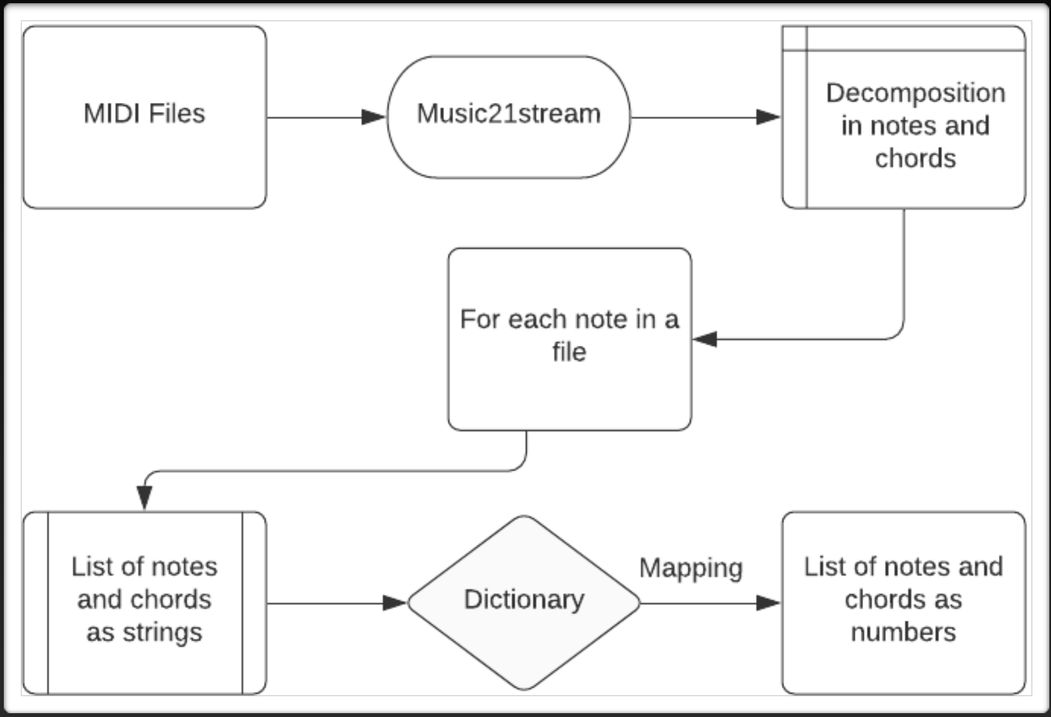
We have used **Google Colab Pro** which gives us priority access to its GPUs



Data Gathering

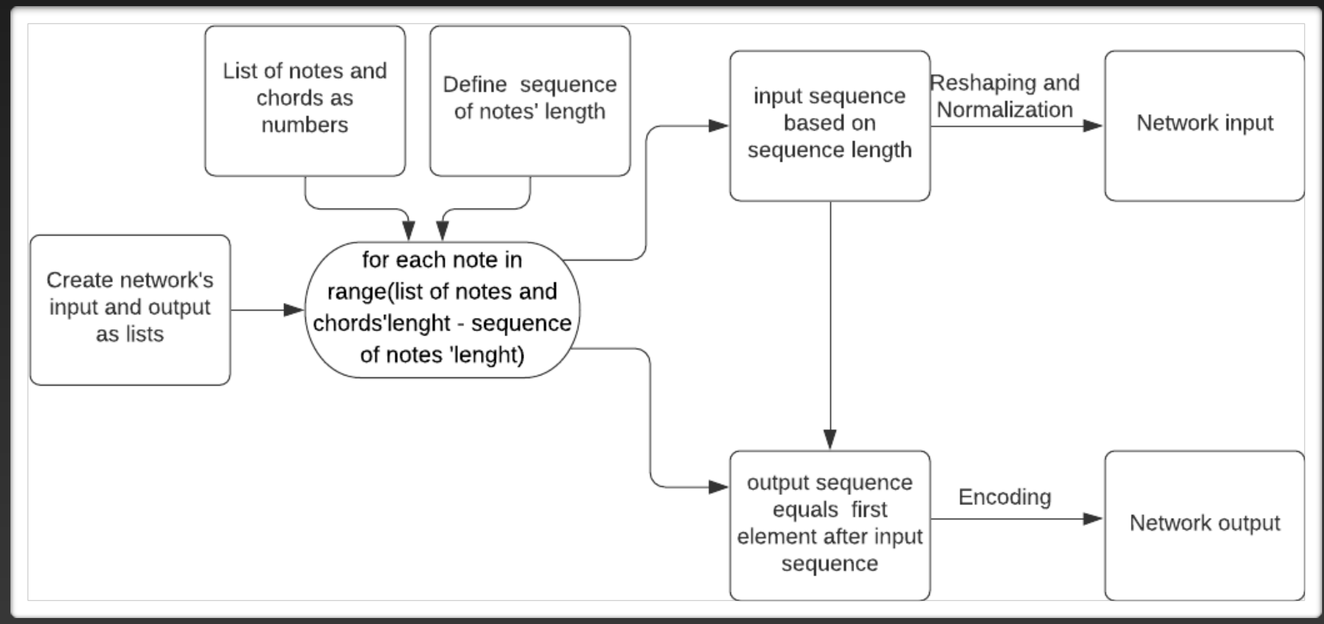
- We decided to use 98 MIDI files of **Mozart**, **Beethoven** and **Chopin**. All these files are parts of piano-solo compositions we found online at www.piano-midi.de.
- We were looking for a **common melodic style**, to make our prediction perform better.
- Despite Mozart and Beethoven belong to different styles, ranging from "classical" to "romantic" period, we have understood they **share a similar musical stamp**, especially in piano-solo compositions.
- The choice of **Chopin** has occurred because we were looking for a third composer who could increase the size of training set and in the meanwhile keeps a similar identity to the former artists based on longer lines of melody, gradually increasing and decreasing.

Pre-processing



1. The first main point is the **parsing** of notes, chords and pitches into an array.

2. Then we created a sorted **set** to keep unique notes, which are then mapped in a dictionary to make them numerical values.



3. Our network will use the previously **100 notes** to predict the following one.

4. After created input and output sequences we can **normalize** and **reshape** input data and **encode** the output.

Model Architectures

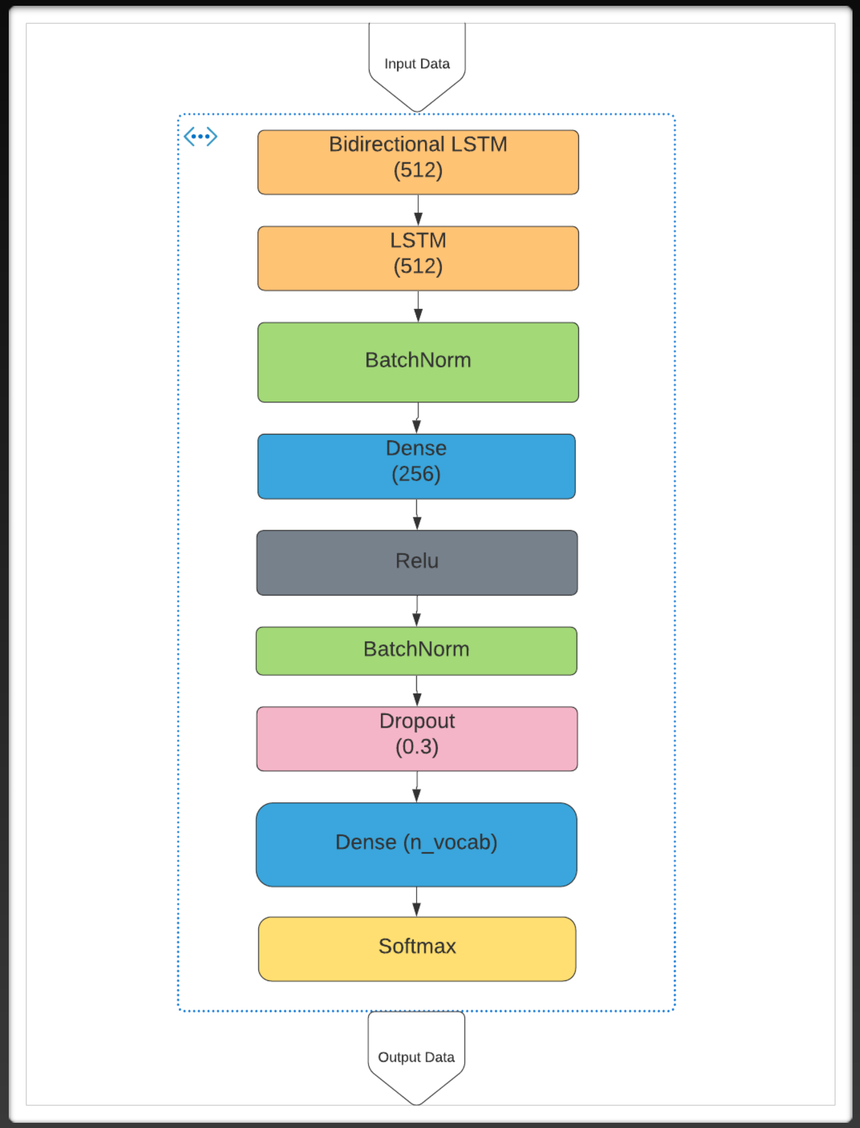
- 4 different macro models, each on update and slightly adjusted.
- All the models have been computed using a validation split of 0.3.
- All the models have been tested using an instance of the same GPU (Tesla P100 PCIE 16Gb).
- As callbacks we used both TensorBoard and traditional callback by Keras.
- We decided to focus more on loss than accuracy.
- Most of our starting LSTM layers have been tested also in Bidirectional way.

Model Architecture

Model 0

- For this first model we tried to mix something related to the State of The Art and something of our knowledge.

Epochs	**50 (+20)**
Optimizer	Rmsprop
Loss	Categorical Crossentropy
Batch Size	128
Avg Loss/epoch	5.40

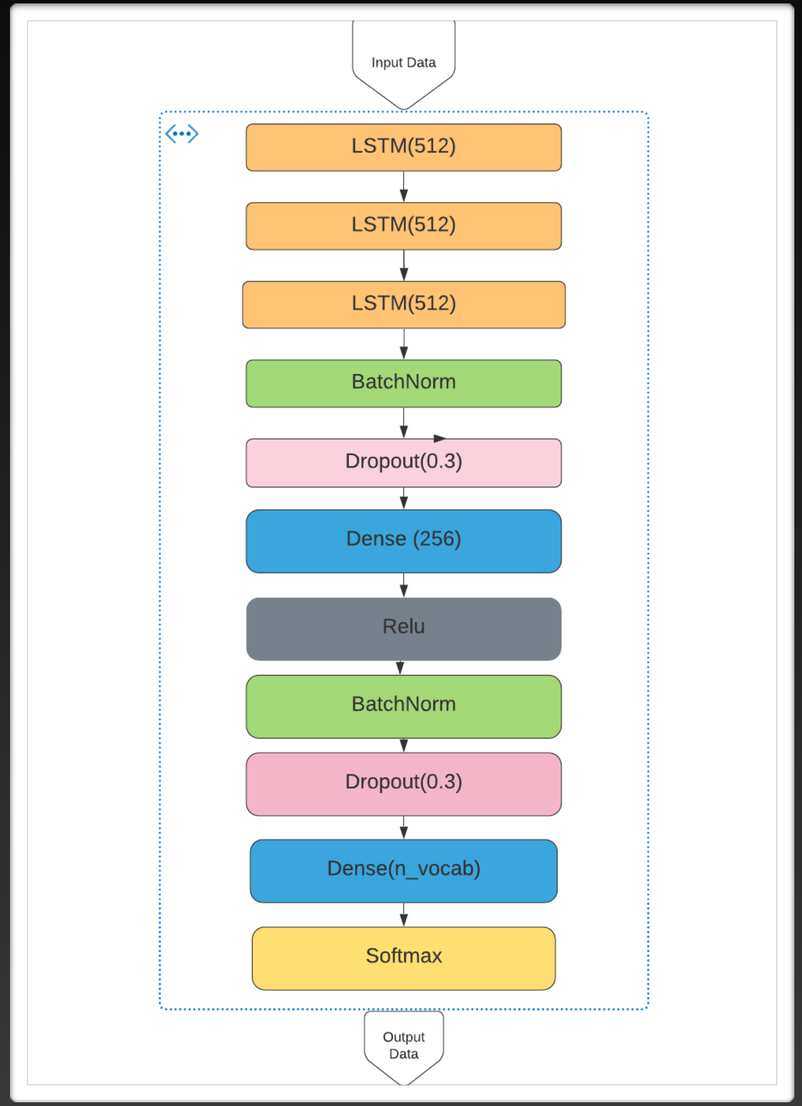


Model Architecture

Model 1

- Ascertained the fact that our previous model was overfitting, we decided to test another model before proceding with the complexity reduction.

Epochs	**90**
Optimizer	Rmsprop
Loss	Categorical Crossentropy
Batch Size	128
Avg Loss/epoch	Fluctuating around 5.75



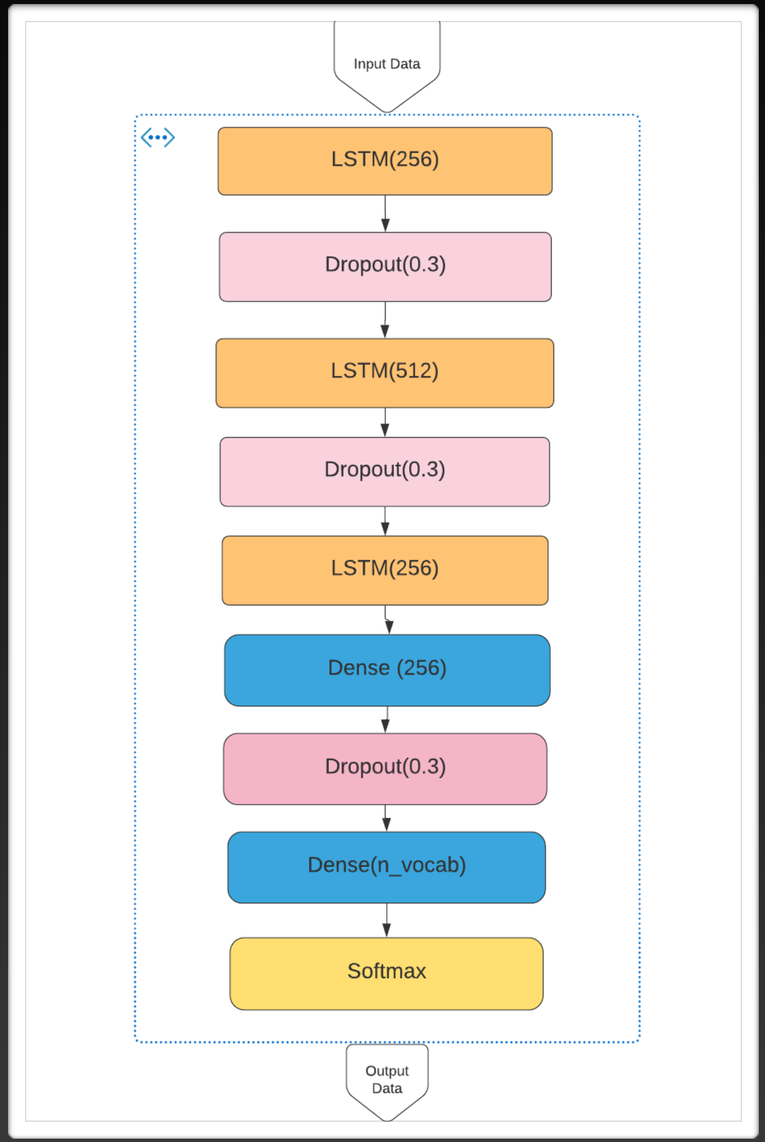
Model Architecture

Model 1b

- We decided to adjust our initial part and to remove the middle activation. We created 2 **LSTM-Dropout** blocks with 256 and 512 neurons respectively.

Epochs	**18**
Optimizer	Rmsprop
Loss	Categorical Crossentropy
Batch Size	128
Avg Loss/epoch	From ep. 6 increases up to 4.83

- Here we started to use another dataset (for computational problems) composed of around 10-15 MIDI less.

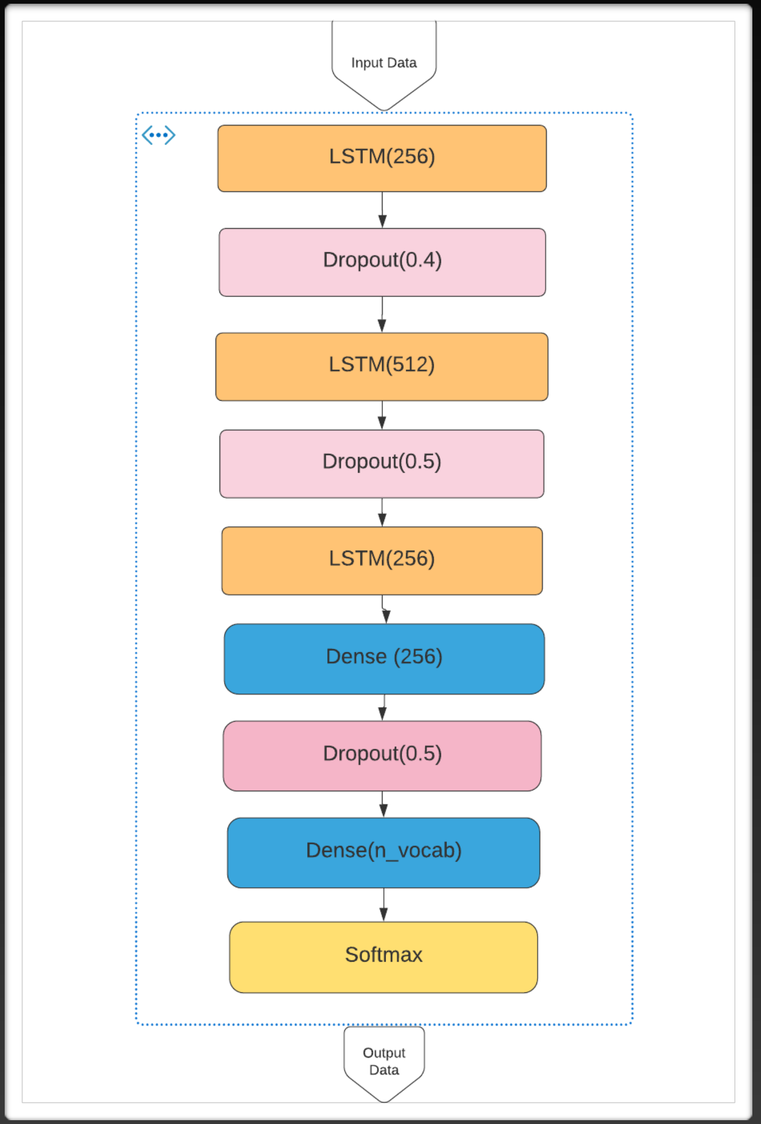


Model Architecture

Model 1c

- In this adjustment of the previous model we tried to play with the dropout layers and increasing their probability. It has not been a good idea since validation loss starts to arise from the beginning.

Epochs	**18**
Optimizer	Rmsprop
Loss	Categorical Crossentropy
Batch Size	128
Avg Loss/epoch	From ep. 3 increases up to 5.53

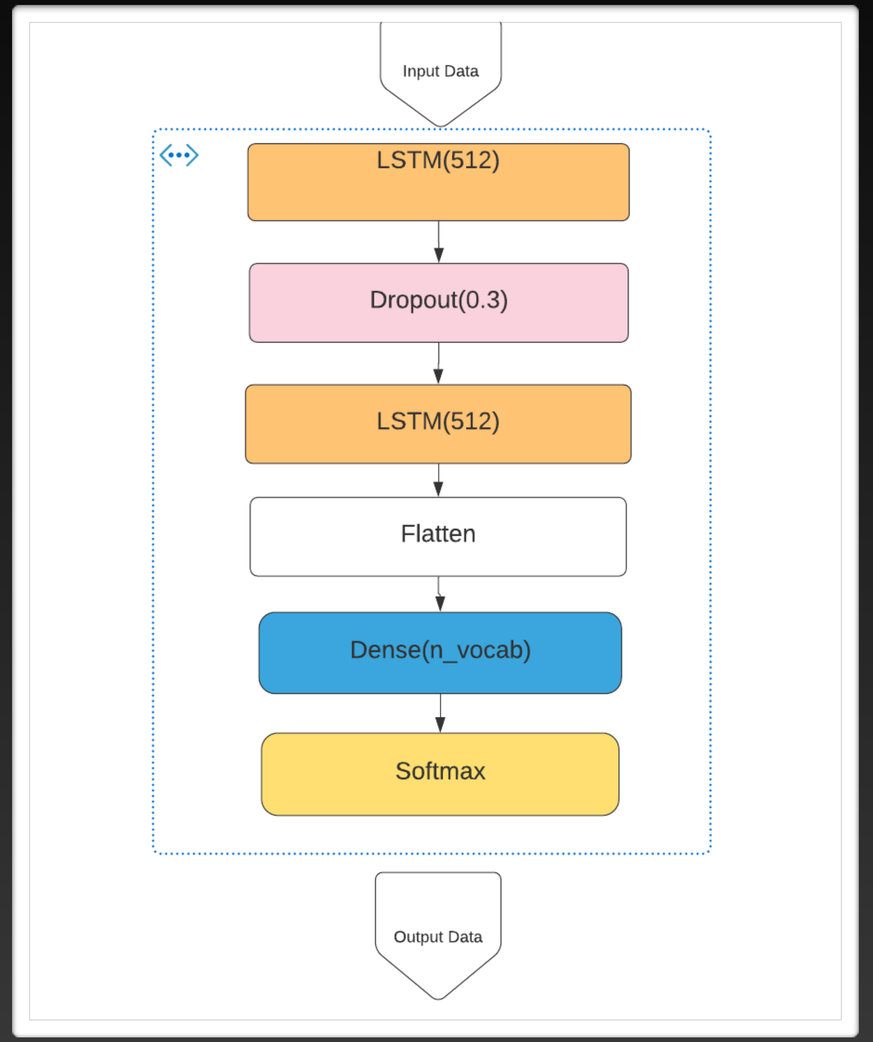


Model Architecture

Model 2

- Since the previous results were quite bad we decided to reduce the complexity of our model. We used the flatten to better prepare our data for the following dense layer.

Epochs	**16**
Optimizer	Rmsprop
Loss	Categorical Crossentropy
Batch Size	128
Avg Loss/epoch	From ep. 4 fluctuating around 4.83

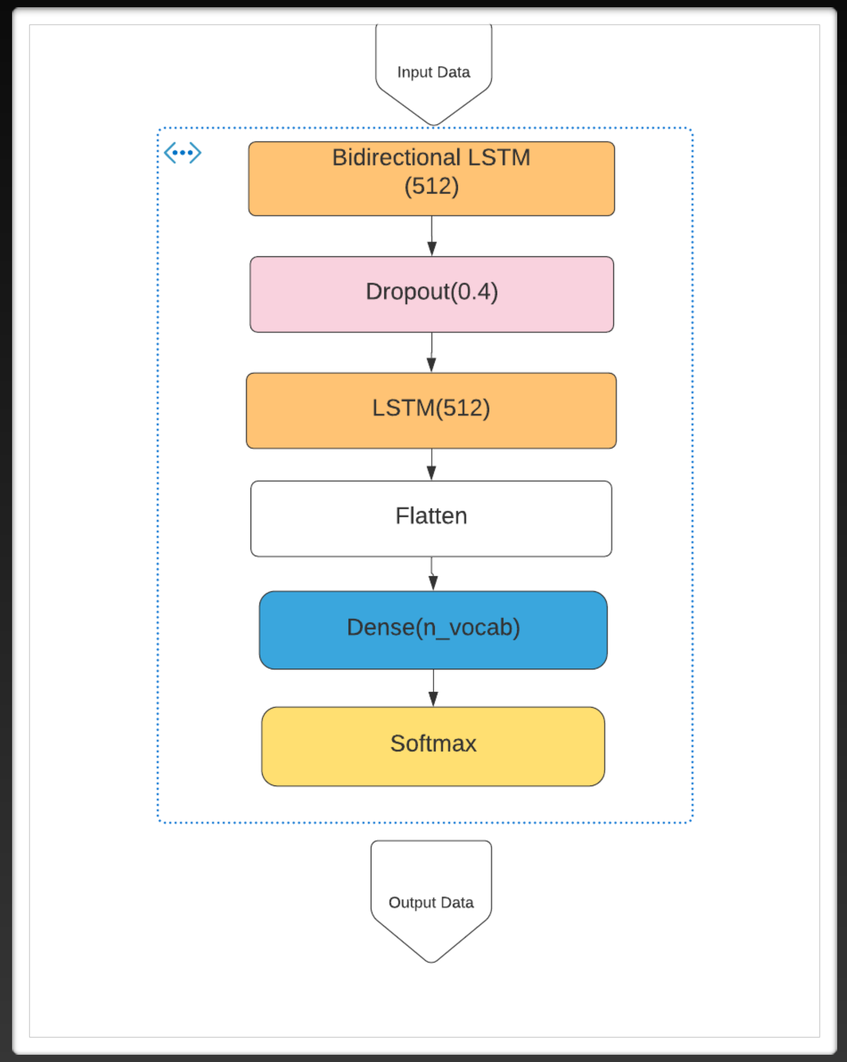


Model Architecture

Model 2b

- First we used again a Bidirectional LSTM as first layer and then we used **Stochastic Gradient Descend** as optimizers.
- We tried also to reduce the batch size from 128 to 64 in order to add some small noise around the model and use it as a kind of regularization.

Epochs	**50**
Optimizer	Rmsprop - SGD
Loss	Categorical Crossentropy
Batch Size	128 - 64
Avg Loss/epoch	Fluctuating around 4.80



Attention

- From model 3 we started to use Attention.
- It makes our network able **to remember the intermediate calculations**.
- It's a variation of the traditional seq2seq architecture. In the seq2seq idea an **encoder** compresses the information into a vector of fixed lenght which can be used as abstraction of the entire input sequence. A **decoder** instead, initializes itself based on that sequence. In this way we're not able to remember longer sequences.
- Using Attention we don't discard the intermediate layers in order to create the vector.

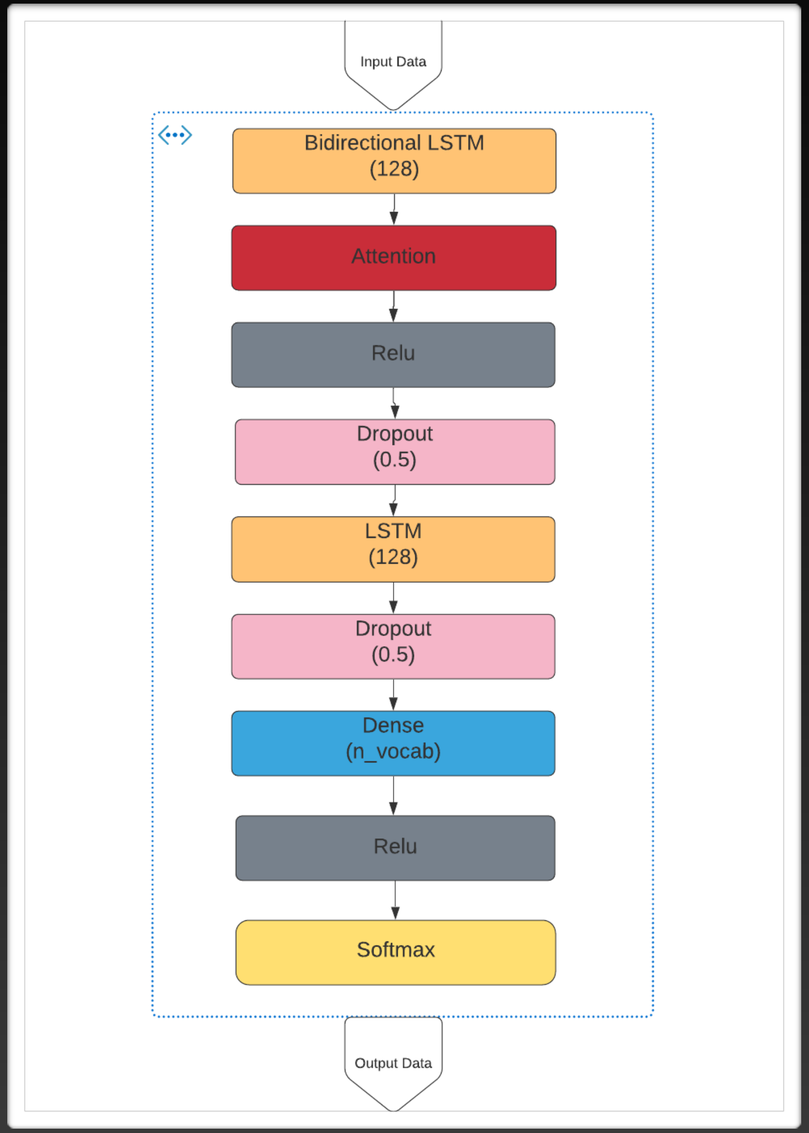
https://pypi.org/project/keras-self-attention/

Model Architecture

Model 3

- We started to use Adam optimizer since it can be seen an improvement of the adaptive Rmsprop.
- As activation of Attention we used both ReLU and softmax.
- We tried also the reduce the number of neurons of the LSTM layers.

Epochs	**18**
Optimizer	Adam
Loss	Categorical Crossentropy
Batch Size	128
Avg Loss/epoch	Fluctuating around 4.84

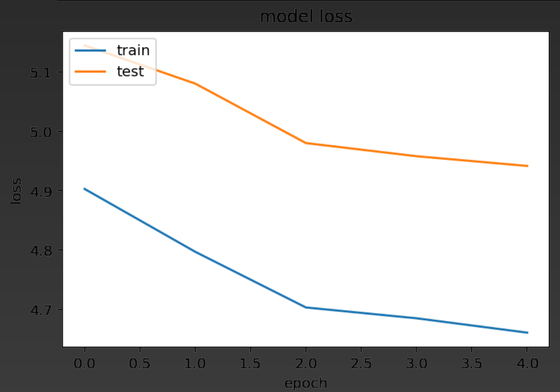


Model Architecture

Model 3b

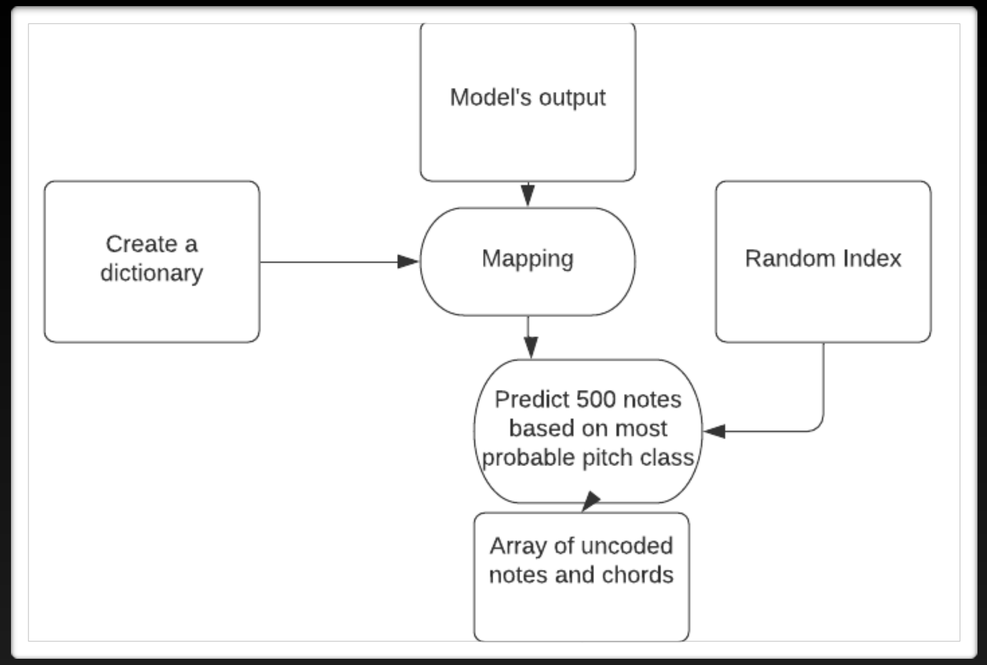
- Understood the real importance of the attention mechanism we tried to compute a smaller model based again on this framework.

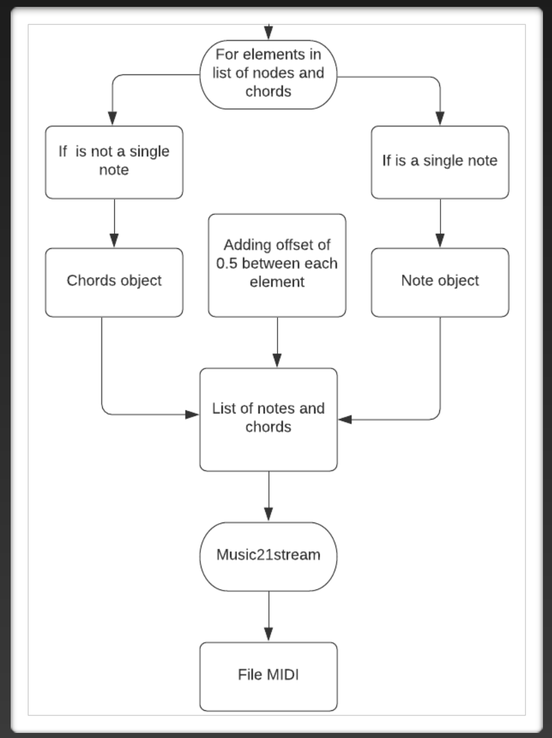
Epochs	**18 (+4)**
Optimizer	Adam
Loss	Categorical Crossentropy
Batch Size	128
Avg Loss/epoch	Fluctuating around 4.84



Monitoring and Post Deployment

- We designed an **ideal** way to deal with **Concept Drift**: a scenario which makes the performance of the model change when it is asked to make predictions on new data that was not part of the training set.
- We thought to use a similarity metrics to understand any kind of difference between the training prediction and a general inference prediction. Similarity are measured in the closed range from 0 to 1.
- We decided to use the **Wasserstein distance** which aims is to measure the minimal effort required to go from one probability distribution to another. The area we'll obtain is the distance between the two curves.
- Re-update the model frequently and allow only single instruments MIDI (piano)





Post Processing and Conversion

1. It's essentially the opposite of Pre procesing part. All start with the creation of a dictionary to map back all integers to pitches-notes-chords.

2. We made many attempts and we defined a general number of notes prediction. The used dimensions are around 250 notes and 500.

3. We made a prediction based on if the element is single (note) or not (chord) and add pitch and offsets.

We have implemented Fluid Synth and SoundFont on a VPS with Ubuntu 18.04.

This will help us to render audio to file and create a new WAV from the MIDI. (All the Python libraries are obsolete)

Conclusions

- Here you can find two of our best prediction.
- We are not so satisfied of the results we got. The performances of our models are not that astonishing probably because of the machine we used to train them.
- Since it has enjoyed us working on it, in the future we will keep working on.

Possible things to do in the future:

- try to increase the sequence lenght in order to understand if this will change our accuracy;
- try to use different layers such as 1D Convolutional layers or a GAN;
- try to increase the number of musical instruments;