# Recurrent harmonizer

José Pablo Ortiz

May 3, 2022

## Introduction

In this work a deep learning model, many-to-many recurrent neural network, will be developed to harmonize a melody in the style of Bach. For this end, the Bach 2/3 part inventions are gonna be used as the data to train the model; it's the first time a recurrent music generation model has been used to fit this data. Most of the previous work that has been used to model Bach's music use the chorales. In the first part we are gonna introduce the context and characteristic of the inventions along with the pianoroll notation. Then, in the second part a short yet concise exposition of the model will be outlined. In the final part, the details of the training, the model selection and the results will be shown.

## Data

The inventions by Johann Sebastian Bach are a 30 piece collection for the keyboard. They are composed as 2 or 3 voice polyphonic pieces. A 2/3 voice polyphonic piece is consists of 2/3 independent melodies, with the property that when played together they sound harmonious. Figure 1 shows the sheet music of one measure of a 2 voice invention. The upper bunch of notes represent the first melody and the lower the second one. The sheet music of all the inventions was obtained from Musescore and then hand preprocessed in order to make the data extraction process more uniform. The information was extracted using the python library MUSIC21. MUSIC21 is an amazing python library with a series of tools 'for helping scholars and other active listeners answer questions about music quickly and simply'

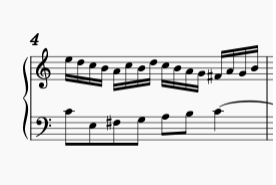

Figure 1: A two part invention

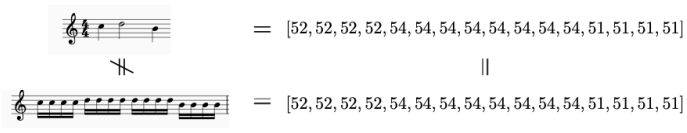


Figure 2: An example on how the piano roll misses some information about a melody.

### Piano Roll

The data format that it is used in this project is the piano roll notation. In the context of music generation using deep learning it was first used by Boulanger-Lewandowski[1]. To represent a sheet music of a 1 voice melody as a piano roll, first it is need to establish the smallest time duration of every note or rest in the sheet music. Then every piece can be seen as finite time discrete sequence of notes; where every time instant has the duration of the smallest note duration. Representing

every note as one of the 88 keys in a keyboard and a rest by the number $-1$. This yields, that a 1 voice melody piece can be seen as sequence $\{v_t\}_{t=1}^{D}$ where $v_t$ is the number of note played at time $t$. The piano roll notation misses some information about the music. Let's suppose that the smallest note duration of a one voice melody is $\frac{1}{16}$ and that there exists a $k$ and $n$ such that $v_k = v_{k+1} = \cdots = v_{k+n}$; then the piano roll notation will not be able to gather information of whether the note $v_k$ is played $n$ times or has duration $\frac{n}{16}$. An example of this it is shown in Figure 2.

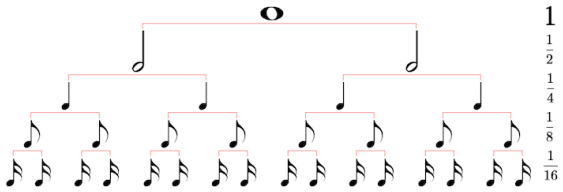


Figure 3: Diagram explaining the duration of notes.

For the dataset in this work we will only take inventions in which the smallest note duration is $\frac{1}{16}$. A two voice invention following the piano roll notation can be written as

$$I = \left\{(v_t^1, v_t^2)\right\}_{t=1}^{D}$$

where $v_t^i$ represent the note a time $t$ of the $i$-th voice and $D$ the duration of the invention. Thus, each time $t$ represents $\frac{1}{16}$ note and the duration of the invention is $\frac{D}{16}$. In the other hand, a three voice invention can be written as

$$I = \left\{(v_t^1, v_t^2, v_t^3)\right\}_{t=1}^{D}.$$

Nonetheless, this work will focus on adding a another voice to a melody. So, in order to use the data of the 3 voice inventions we will extract two 2 voice inventions for every 3 voice invention in the following way:

$$I_1 = \left\{(v_t^1, v_t^2)\right\}_{t=1}^{D} \quad I_2 = \left\{(v_t^2, v_t^3)\right\}_{t=1}^{D}.$$

## Data Augmentation

We can do data augmentation by adding some constant to every note. Because there are 12 tones, in order to not be redundant in a musical way, it is only possoble to add 12 consecutive integers $\{-6, -5, \ldots, 6\}$. Thus, by doing this data augmentation the dataset has 11 more observations per observation.

| 2 voice Invention | Number of different notes | Number of different notes with data augmentation |
|---|---|---|
| voice 1 | 34 | 45 |
| voice 2 | 39 | 50 |

Table 1

| 3 voice Invention | Number of different notes | Number of different notes with data augmentation |
|---|---|---|
| voice 1 | 27 | 39 |
| voice 2 | 33 | 45 |
| voice 3 | 39 | 50 |

Table 2

| 2 and 3 voice Invention | Number of different notes | Number of different notes with data augmentation |
|---|---|---|
| voice 1 | 37 | 48 |
| voice 2 | 47 | 59 |

Table 3

## One Hot Encoding

The one hot encoding consists in assigning every note used in an invention a canonical vector of dimension the size of notes used in an invention. Given Table 3 every note in voice 1 will be represented as a canonical vector of dimension 48 and voice 2 as a canonical vector of dimension 59. From now on, the sequence $\{v_t\}_{t=1}^{D}$ will be the one hot encoding of a melody; that is, a sequence of canonical vectors. Another advantage of using the one hot encoding is that we can visualize a whole piece as shown in Figure 4 where a one hot encoding was taken with a number of notes of 88.

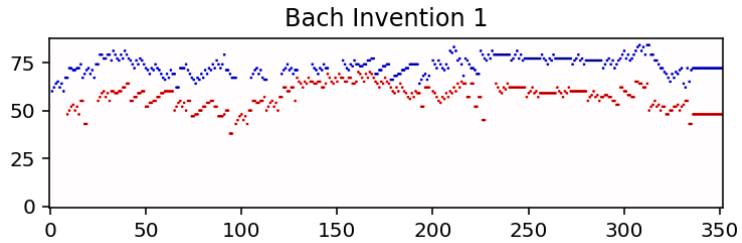


Figure 4: Visualization of Bach's invention no. 1

## Rolling Window

The model that is gonna be trained is a many-to-many deep recurrent neural network of size $T$. In order to train the model it is necessary to obtain all the rolling windows of size $T$ and stride $s$ of the inventions. The rolling windows of size $T$ and stride $s$ of an invention are

$$\left\{ \left[ (v^1_{ks+t}, v^2_{ks+t}) \right]_{t=1}^{T} \mid \quad \text{for } k = 0, 1, \ldots, \frac{D-T}{s} \right\}.$$

The aim of this project is to create a model that given a melody of high notes it adds a second melody of bass notes so that when played together they sound harmonious. For this end, it's necessary to have information about the current melody of voice and the bass melody. So the input of the model will have the form

$$X = \begin{pmatrix} X_{11} & X_{12} & \ldots, X_{1T} \\ X_{21} & X_{22} & \ldots, X_{2T} \end{pmatrix} = \begin{pmatrix} v^1_{ks} & v^1_{ks+1} & \ldots & v^1_{ks+T} \\ v^2_{ks-1} & v^2_{ks} & \ldots & v^1_{ks+T-1} \end{pmatrix}, \tag{1}$$

and the output

$$Y = \begin{pmatrix} Y_1 & Y_2 & \dots, Y_T \end{pmatrix} = \begin{pmatrix} v_{ks}^2 & v_{k+1}^2 & \dots & v_{ks+T}^2 \end{pmatrix}. \tag{2}$$

# Recurrent Harmonizer

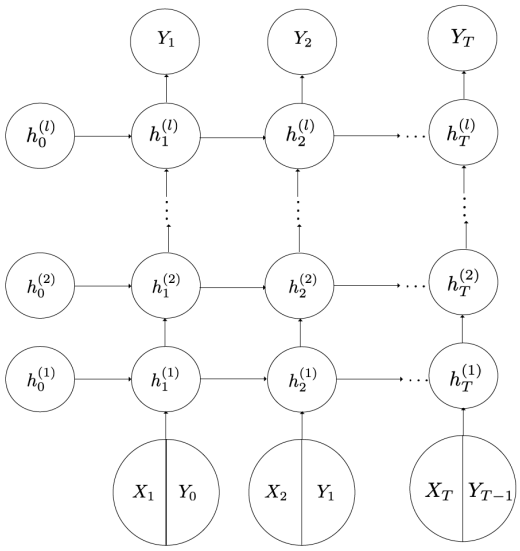


Figure 5: Diagram explaining the duration of notes.

## Model

The many-to-many RNN is probabilistic generating model, it is a generative because the task is to predict the conditional probability mass function

$$p(y_t|y_{t-1}, \dots, y_0, x_t, \dots, x_1). \tag{3}$$

As mentioned earlier a many-to-many deep RNN of size $T$ consists of the following discrete time recurrence equations

$$\begin{aligned} h_t^{(1)} &= g\left(W^{(1)}h_{t-1} + U_1^{(1)}(X_{1t}) + U_2^{(1)}(X_{2t})\right) \\ h_t^{(2)} &= g\left(W^{(2)}h_t^{(1)} + U^{(2)}h_{t-1}^{(2)}\right) \\ &\vdots \\ h_t^{(l)} &= g\left(W^{(l)}h_t^{(l-1)} + U^{(l)}h_{t-1}^{(l-1)}\right) \\ \hat{p}(y_t|y_{t-1}, \dots, y_0, x_t, \dots, x_1) &= \hat{p}(y_t|h_t^{(1)}, \dots, h_t^{(l)}) = \text{softmax}(Vh_t^{(l)}), \end{aligned} \tag{4}$$

for $t = 1, \dots, T$. By abusing notation $g$ represents a Long Short Term Memory cell, $X$ and $Y$ are the input and output as in (1) and (2), $l$ the depth of the neural network, and $h_t^{(\cdot)}$ the hidden states. The entries of the matrices $W^{(\cdot)}, U^{(\cdot)}$ and $V$ are the weights/coefficients that are up to estimation. The dimension of this matrices are known as the number of neurons in a layer and they are hyperparameters of the model. Because we want to predict a probability, the function softmax it's used as the last activation of the RNN. The idea behind this model is that the hidden

space $h_t^{(i)}$ has the information about the process up until time $t$; in a more mathematical way, that is $h_t^{(i)} \subset \sigma(Y_0, \ldots, Y_{t-1}, X_1, \ldots, X_t)$ where $\sigma(Y_0, \ldots, Y_{t-1}, X_1, \ldots, X_t)$ is the sigma algebra generated by the random variables $(Y_0, \ldots, Y_{t-1}, X_1, \ldots, X_t)$[2].

### Inference or second voice generation

Once the model is trained and given a melody $X = [X_1, \ldots, X_T]$ and an initial value $Z_0$ it is possible to do inference(or creating a second voice) by following the recursion

$$\begin{aligned} Z_1 &= \arg\max\{p(Z_1|Z_0, x_1)\} \\ Z_2 &= \arg\max\{p(Z_2|Z_1, Z_0, X_1, X_2)\} \\ &\vdots \\ Z_T &= \arg\max\{p(Z_t||Z_{T-1}, \ldots, Z_0, X_1, \ldots, X_T))\} \end{aligned} \tag{5}$$

### Hyperparameters of the model

$l$ represents the depth or number of layers in a neural network. Let $c_1$ and $c_2$ be the number of notes in voice 1 and 2, then the matrices $U_1^{(1)} \in \mathbb{R}^{c_1 \times n_1}$ and $U_2^{(1)} \in \mathbb{R}^{c_2 \times n_1}$ where $n_1$ is the numbers of neurons in the first layer. For $i = 2, \ldots l$ we have that $U^{(i)} \in \mathbb{R}^{n_{i-1} \times n_i}$; meanwhile $W^{(i)} \in \mathbb{R}^{n_i \times n_i}$. The hyperparameter $n_i$ represents the number of neurons in the $i$-th layer. The trade off of this hyperparameters is that more depth and more neurones per layer can offer more flexibility but can overfit the data and make it harder to train the model.

## Training and model selection

To train the model first we need set the size and the stride of the rolling window as in (1) to generate the training data $X$ and $Y$. We will set $T = 32$ and $s = 1$ to generate the training data to obtain a sample of 52284 rolling windows. This 52284 were split in an $n = (80\%)$ training set and $n_{\text{val}} = (20\%)$ validation set. We acknowledge that the model selection

### Loss function and numerical minimization

Because we want to make inference on a probability function the loss function that is gonna be used is the cross entropy; that is:

$$\frac{1}{n}\sum_{i=1}^{n} L_i = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{T}\sum_{t=1}^{T}\log\left(\hat{p}(y_t|h_t^{(1)}, \ldots, h_t^{(l)}))\right)\right], \tag{6}$$

where $L_i$ is the loss of each observation. Thus, in order to estimate the weight matrices we need to minimize (7). The algorithm that it's used to minimize(6) is minibatch stochastic gradient descent with the gradient, calculated by backpropagation, being updated using the ADAM method.

Table 4 shows the different metrics to judge the performance of our different models. For our final model we are gonna select the one with the validation accuracy closest to 1. That is the one with $l = 2$ and each layer with 512 neurons.

### Results

In order to show the results of the model we will use the prelude from Bach cello suite no 1. The piano roll of the result is shown in Figure 6

| layers | hidden | train loss | train accuaricy | val_loss | val_sparse_categorical_accuracy |
|---|---|---|---|---|---|
| 1 | 32 | 1.40 | 0.59 | 1.40 | 0.59 |
| 1 | 64 | 1.22 | 0.64 | 1.23 | 0.64 |
| 1 | 128 | 0.91 | 0.73 | 0.92 | 0.72 |
| 1 | 256 | 0.44 | 0.88 | 0.47 | 0.87 |
| 1 | 512 | 0.20 | 0.94 | 0.25 | 0.93 |
| 2 | 64,64 | 1.11 | 0.67 | 1.12 | 0.67 |
| 2 | 128,128 | 0.62 | 0.83 | 0.64 | 0.82 |
| 2 | 256,256 | 0.25 | 0.93 | 0.29 | 0.92 |
| 3 | 64,32,64 | 1.23 | 0.64 | 1.23 | 0.64 |
| 3 | 128,64,128 | 0.74 | 0.79 | 0.76 | 0.78 |
| 3 | 256,128,256 | 0.29 | 0.92 | 0.33 | 0.91 |

Table 4: The metrics of all the trained models

Hearing the results, it seems that the model has success in creating a Bach sound. Nonetheless, it's not able to capture the information about the long term structure of the piece.

# References

[1] Boulanger-Lewandowski, N., Bengio, Y.,& Vincent, P.(2012). *Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription.* ICML 2012: Edinburgh, Scotland.

[2] Calin, O. (2020). *Deep Learning Architectures: A Mathematical Approach* (1st ed.). Springer.

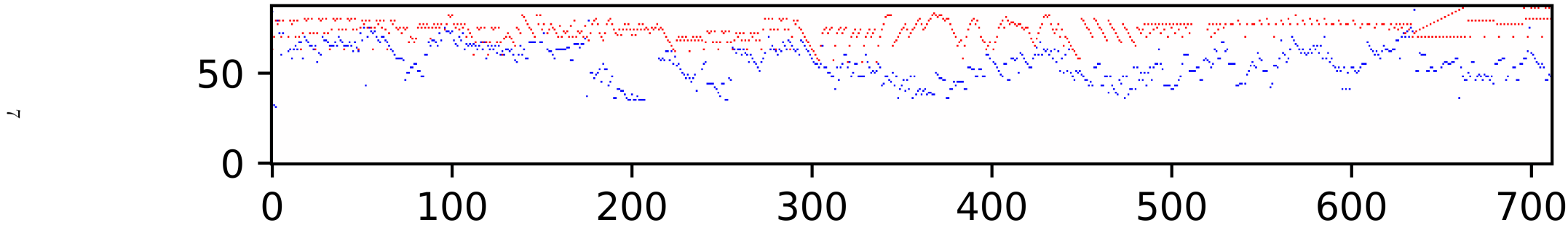


Figure 6: Diagram explaining the duration of notes.