UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Posteriograms Postprocessing for Multi-Pitch Estimation

Author:
Yurii ANTENTYK

Supervisor:
Oles DOBOSEVYCH

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

I, Yurii ANTENTYK, declare that this thesis titled, “Posteriograms Postprocessing for Multi-Pitch Estimation” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Posteriograms Postprocessing for Multi-Pitch Estimation

by Yurii ANTENTYK

Abstract

Music Transcription is a task of converting a musical recording into sheet music for further reproduction. The problem is still unsolved and requires a high level of expertise. Most of the works split the task into several subproblems. First of them is called *frame-level transcription*, which predicts the set of fundamental frequencies in the original recording for every frame. This subproblem is the main focus of this work.

The solution to *frame-level transcription* is called a *piano-roll representation* - a binary matrix which represents whether the given note has been played in the frame or not. However, most of the approaches do not produce a *piano-roll representation* in an end-to-end fashion. They rather output a *posteriogram* - real matrix with the same dimensions, which represents the level of uncertainty of whether note has been played during the frame. Ycart and Benetos, 2018 shows that *Long Short Term memory network* can be trained to post-process the *posteriograms* and improve the *piano-roll* representation instead of simply cropping the *posteriogram* at some value.

In this work, we train more robust *LSTM* network and experiment with different types of posteriograms.

Acknowledgements

I would like to thank my beloved parents, friends and colleagues who supported me through all the difficulties. Special gratitude goes to the Faculty of Applied Sciences and personally to Oles Dobosevych for interesting ideas, computing power and valuable pieces of advice.

Contents

Declaration of Authorship **i**

Abstract **ii**

Acknowledgements **iii**

1 Introduction **1**
1.1 Problem Description 1
1.2 Motivation 1
1.3 AMT Subproblems 1
1.4 Raw Audio Preprocessing 3
1.4.1 Fourier Transform Related Methods 3
1.4.2 Harmonics 3

2 Related Works **5**
2.1 Spectrogram Factorization 5
2.2 Neural Networks 6
2.3 Mixed Approaches 6

3 Posteriogram Posprocessing **7**

4 Model **9**
4.1 Classification Task 9
4.2 Model Architecture 9
4.3 Posteriogram Types 10

5 Dataset **11**

6 Metrics **12**
6.1 Precision 12
6.2 Recall 12
6.3 F-score 13
6.4 Accuracy 13

7 Experiments **14**
7.1 Data 14
7.2 Training 14
7.3 Loss Function 14
7.4 Experiments 15
7.5 Results 15
7.5.1 Comparison Table 16
7.5.2 Experiment 1 16
7.5.3 Experiment 2 17
7.5.4 Experiment 3 18

8 Results **19**
8.1 Conclusions . . . 19
8.1.1 CQT Posteriograms Results . . . 19
8.1.2 Low Recall . . . 19
8.2 Further Work . . . 20
8.2.1 Fading and Sustain Pedal . . . 20

Bibliography **21**

List of Figures

1.1 AMT subproblems possible results illustrated on an extract from J.S. Bach: Prelude and Fugue in C major, Well Tempered Clavier, book 1 . 2
1.2 Comparing different instruments spectrum playing the same pitch (A4). On the left: piano. On the right: ukulele. Ukulele spectrum is scaled in order for the peaks to match in amplitude. 3
1.3 Aliasing illustration. 4500-hz and 5500-hz cosine signals are indistinguishable when sampled at 10000 fps. The figure is an adaptation from Downey, 2016 4

2.1 *NMF* with $R = 5$ performed on an extract from J.S. Bach: Prelude and Fugue in B-flat major, Well Tempered Clavier, book 1. Performed by Andras Schiff. 5

3.1 Analyzing an extract from J.S. Bach: Prelude and Fugue in C major, Well Tempered Clavier, book 1 8

4.1 *RNN* cell. Prediction for time stamp t is based on previous hidden state h_{t-1} and current input x_t 9
4.2 *LSTM* cell. Prediction for time stamp t is based on previous hidden state h_{t-1}, previous cell state c_{t-1} and current input x_t. Cell state is changed via "update" and "forget" gates. The image is an adaptation from wikipedia 10

7.1 *BCE* and focal loss comparison. 15
7.2 Experiment 1. From top to bottom: input posteriogram, LSTM activations, prediction piano-roll, actual piano-roll 16
7.3 Experiment 2. From top to bottom: input posteriogram, LSTM activations, prediction piano-roll, actual piano-roll 17
7.4 Experiment 3. From top to bottom: input posteriogram, LSTM activations, prediction piano-roll, actual piano-roll 18

8.1 Recall curve for *Experiment 3*. 19

List of Tables

7.1 Experiments configurations 15
7.2 Experiments best train/test framewise metrics 16

List of Abbreviations

AMT	**A**utomatic **M**usic **T**ranscription
MIR	**M**usic **I**nformation **R**etrieval
MPE	**M**ulti-**P**itch **E**stimation
FT	**F**ourier **T**ransform
STFT	**S**hort **T**ime **F**ourier **T**ransform
CQT	**C**onstant **Q** **T**ransform
NMF	**N**on-**N**egative **M**atrix **F**actorization
PLCA	**P**robabilistic **L**atent **C**omponent **A**nalysis
RNN	**R**ecurrent **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
RCNN	**R**ecurrent **C**onvolutional **N**eural **N**etwork
CTC loss	**C**onnectionist **T**emporal **C**lassification loss
HCQT	**H**armonic **C**onstant **Q** **T**ransform
LSTM network	**L**ong **S**hort **T**erm **M**emory network
FL	**F**ocal **L**oss

Chapter 1

Introduction

1.1 Problem Description

Automatic Music Transcription (AMT) is a task of converting music signals into a notation that can be used to reproduce the original signal. It is considered to be one of the most important tasks in the field of *Music Information Retrieval (MIR)*. It is still unsolved, with human experts outperforming any solution in the field. However, this task is considered to be of high interest, with lots of approaches and novel methods applications.

1.2 Motivation

Although humans outperform computers in *AMT* task, it still requires a high level of expertise. Not even all musicians are capable of it. So, even imperfect or incomplete solutions to *AMT* can be useful for the people as a tooltip or a guideline. The main motivation for solving this task is improvisation capturing, as there are no musical scores for it.

1.3 AMT Subproblems

Despite some works trying to tackle AMT in an end-to-end fashion (Román, Pertusa, and Calvo-Zaragoza, 2019), most of the works split the tasks into several subproblems (Benetos et al., 2019, see 1.1):

- Frame-level Transcription
- Note-level Transcription
- Stream-level Transcription
- Notation-level Transcription

Frame-level Transcription (also known as *Multi-Pitch Estimation* or *MPE*) is a task of detecting a set of fundamental frequencies at each time-stamp of the input signal (see 1.1a).

Note-level Transcription goes one step beyond the previous one and detects onset and offset time of each note (see 1.1b).

Apart from the onset and offset time, *Stream-level Transcription* associates each note with a stream. It could be responsible for a particular instrument or voice in the original recording (see 1.1c).

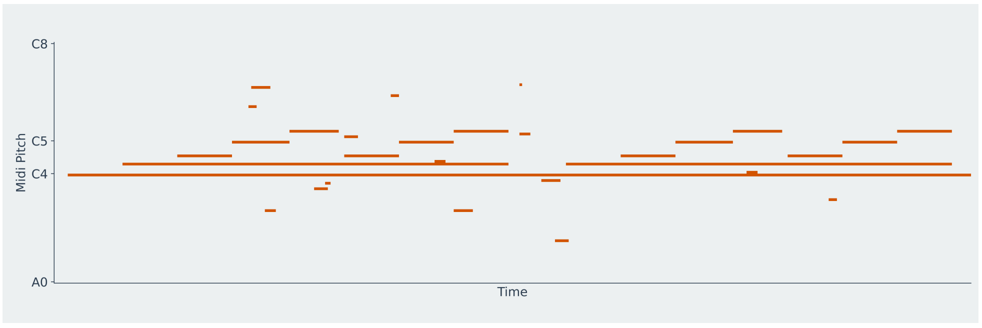
Finally, *Notation-level Transcription* solves AMT task by producing a musical notation from the original recording using notes onsets, offsets and stream information (see 1.1d).

It is worth noting that despite *MPE* seeming the most straightforward task at first, it has lots of peculiarities, such as:

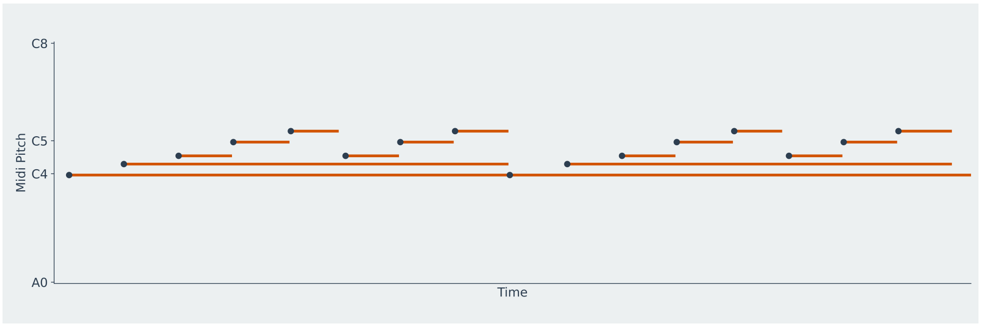
- limited annotated datasets (especially with instrument labels)
- choosing the right audio representation

The problem is also of high importance, because the bad solution to *frame-level* transcription may lead to the error accumulation in the other *AMT* subproblems.

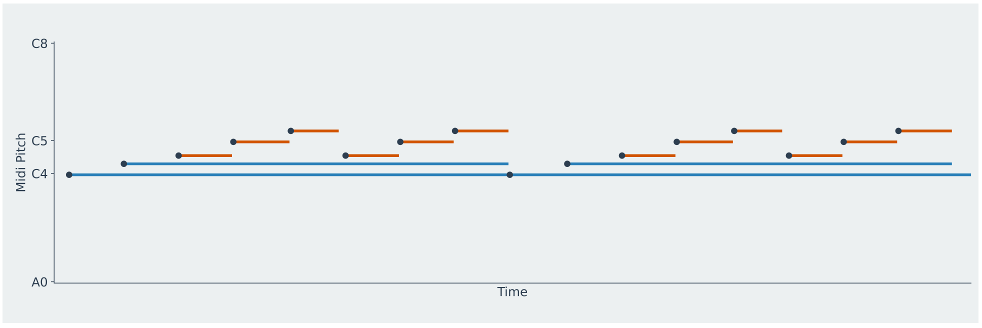
This work is mostly focused on investigating *Multi-Pitch Estimation* for a single piano instrument.



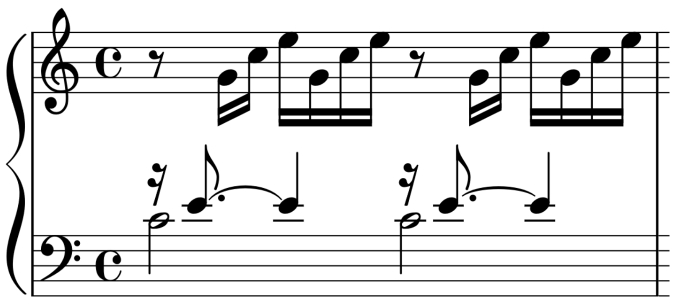
(A) *Frame-level Transcription*



(B) *Note-level Transcription*. Blue dots indicate note offset



(C) *Stream-level Transcription*. Each color represents a separate stream. In this particular example blue stream corresponds to the left hand and orange stream to the right hand respectively.



(D) *Notation-level Transcription*

FIGURE 1.1: AMT subproblems possible results illustrated on an extract from J.S. Bach: Prelude and Fugue in C major, Well Tempered Clavier, book 1

1.4 Raw Audio Preprocessing

1.4.1 Fourier Transform Related Methods

As raw audio is used to represent the sound wave itself (by recording the wave amplitudes at discrete points in time), it is clear, that some data preprocessing should be done as a first step. Most of the works incorporate *Fourier Transform (FT)* technique that decomposes the input signal into a sum of sinusoids with different frequencies thus transforming the input signal from time to frequency domain. Since one FT on the whole audio snippet is not a good representation, the input signal is split into pieces, and *FT* is performed on each piece alone. This is known as *Short Time Fourier Transform (STFT)*. The image obtained by horizontally stacking the *Fourier Transforms* obtained at each timestamp is called a *spectrogram*.

However, there are limitations to this approach: denote δt to be a length of the piece *(time resolution)*, δf to be a difference between adjacent *FT* frequency bins *(frequency resolution)*. According to the *Heisenberg Uncertainty Principle*, $\delta t \cdot \delta f > 1/4\pi$. Meaning that δt and δf cannot be arbitrarily small at the same time.

On the other hand, *STFT* does not consider the fact that piano key frequencies are logarithmically distributed: given that midi pitch p_i has frequency f_i, frequency of the successive pitch p_{i+1} can be calculated by the following formula: $f_{i+1} = f_i \cdot (2^{1/12})$, meaning that the frequency doubles with each octave. Hence, we are interested in logarithmically spaced frequencies with smaller time resolution for high frequencies and bigger time resolution for low frequencies respectively. This issue is resolved in *constant-Q transform*, a variation of *STFT* which maintains constant $(\delta f / \delta t)$ ratio.

1.4.2 Harmonics

When it comes to practice and investigating a *spectrogram*, some of the amplitude coefficients that do not correspond to any of the fundamental frequencies are present in the spectrum. These are called *harmonics*.

Because each instrument has unique sounding, the spectrum of 2 different instruments should vary even if the same note is being played. Fundamental frequency will not be affected, whereas two pieces will have different harmonics, which should be responsible for uniqueness of the instrument timbre.

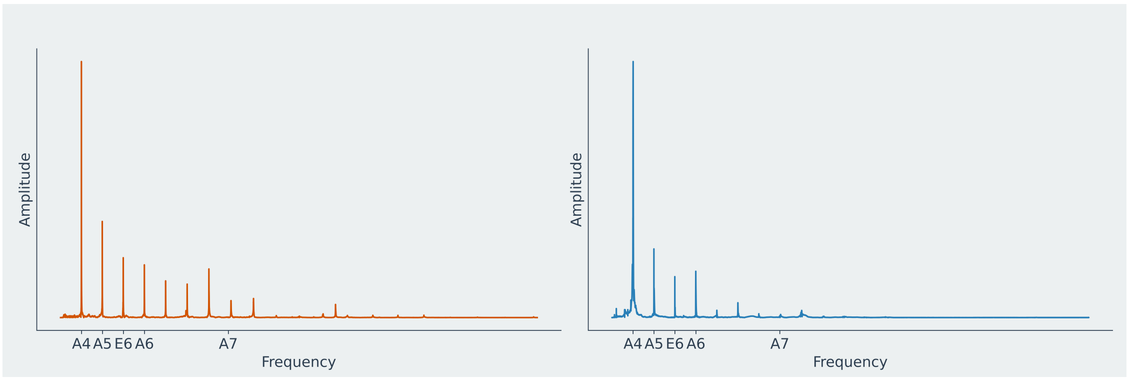


FIGURE 1.2: Comparing different instruments spectrum playing the same pitch (A4). On the left: piano. On the right: ukulele. Ukulele spectrum is scaled in order for the peaks to match in amplitude.

The frequencies of harmonics are often some multiples of the fundamental. Sadly, it is not always true. According to *Nyquist-Shannon theorem*, for an input signal with sample rate $2 \cdot x$ the maximal frequency, that can be restored from the signal is x. It means that if some of the harmonics have a frequency greater than x, it will be indistinguishable from some other low-frequency component. This phenomenon is known as *aliasing*.

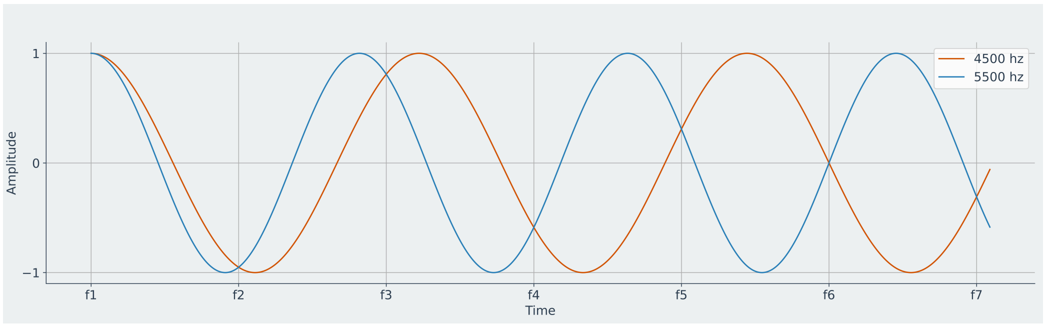


FIGURE 1.3: Aliasing illustration. 4500-hz and 5500-hz cosine signals are indistinguishable when sampled at 10000 fps. The figure is an adaptation from Downey, 2016

Things get even more complicated when there is more than one fundamental frequency. Because of harmonics overlapping, it is even harder to identify the set of fundamental ones.

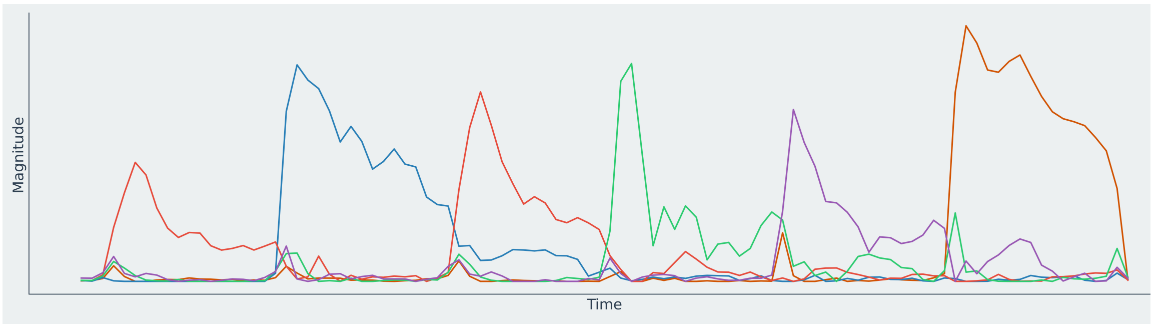
Chapter 2

Related Works

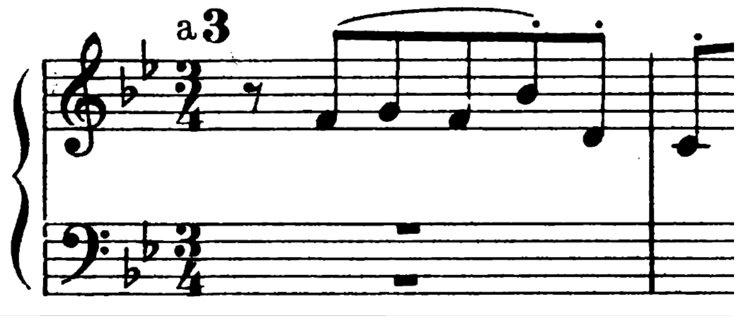
There are two main directions for tackling *MPE*: spectrogram factorization and neural networks.

2.1 Spectrogram Factorization

Most of the works that rely on factorization techniques use *Non-Negative Matrix Factorization* (*NMF*, see Smaragdis and Brown, 2003). It takes a non-negative $M \times N$ spectrogram as input and approximates it as a product of two non-negative matrices: $W \in \mathbb{R}_{+}^{M\times R}$ and $H \in \mathbb{R}_{+}^{R\times N}$ by minimizing a reconstruction error. By setting reasonable R, we can treat this method as a feature extractor, with rows of H being activation coefficients and columns of W being spectral templates, each of which corresponds to a particular note.



(A) Activation coefficients of rows of H. Different color represents a different row.



(B) Corresponding sheet music

FIGURE 2.1: *NMF* with $R = 5$ performed on an extract from J.S. Bach: Prelude and Fugue in B-flat major, Well Tempered Clavier, book 1. Performed by Andras Schiff.

Another approach used for *MPE* is *Probabilistic Latent Component Analysis* (*PLCA*, see Smaragdis, Raj, and Shashanka, 2006). It is a statistical approach that also factorizes spectrogram matrix by modeling it as a mixture of marginal distribution products.

2.2 Neural Networks

With the growth in computing power, more data science methods are being applied to solve the problem of music transcription, such as *Recurrent Neural Networks (RNN)* and *Convolutional Neural Networks (CNN)*. For example, in Román, Pertusa, and Calvo-Zaragoza, 2019 *Convolutional Recurrent Neural Network (CRNN)* with *Connectionist Temporal Classification (CTC)* loss is used on a spectrogram to obtain a musical score and solve *AMT* in an end-to-end fashion. In Liu, Guo, Wiggins, et al., 2018 *CNN* is used for onset/offset and pitch prediction. In Wu, Chen, and Su, 2019 the problem is treated as semantic segmentation, and a U-net like model is used to solve *MPE* task.

2.3 Mixed Approaches

However, there are quite a few works that combine some of the approaches or incorporate knowledge from a musical or audio domain. For example, physical instrument playing limitations are incorporated into *PLCA* to achieve convergence while solving *MPE* for harmonica recordings (Lins et al., 2019). In Bittner et al., 2017 *Harmonic Constant Q Transform (HCQT)* is used as an input representation, which is a several *CQT*s, starting at different multiples of f_0, which correspond to the harmonics. In Ycart and Benetos, 2018 *Long Short Term Memory (LSTM)* network is used to process *NMF* output and improve results of solving *MPE* task. Problems, described in Ycart and Benetos, 2018 will be the main focus of this work.

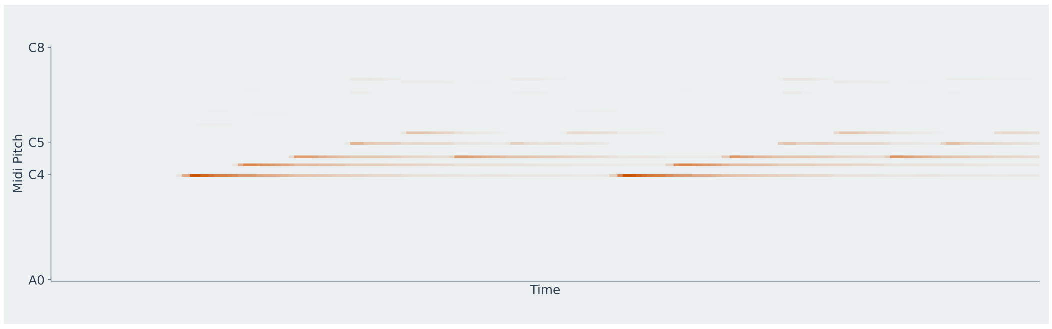
Chapter 3

Posteriogram Posprocessing

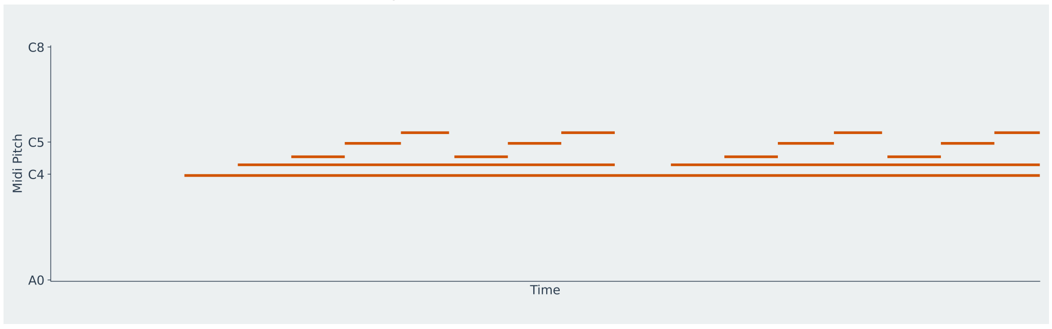
Most of the approaches for tackling *MPE* that were described in the previous chapter have some preprocessing and postprocessing steps. In this work, postprocessing steps are being investigated.

The result of solving the *MPE* problem is a binary matrix $N \times M$ which is called a *Piano-Roll Representation* (see 3.1b), where N is number of notes in the piano (most often 88 keys). M is a number of frames, where each frame is responsible for some short extract of the original recording (also called *resolution* of the track). However, many of the popular approaches do not produce a *Piano-Roll* but rather a *Posteriogram* - a matrix with the same dimensions and real numbers, which represent the level of uncertainty of whether the note has been played in the frame or not (see 3.1a). For example, a *posteriogram* for a neural-network model might be a matrix of real values from 0 to 1, which represent a probability of the note being played.

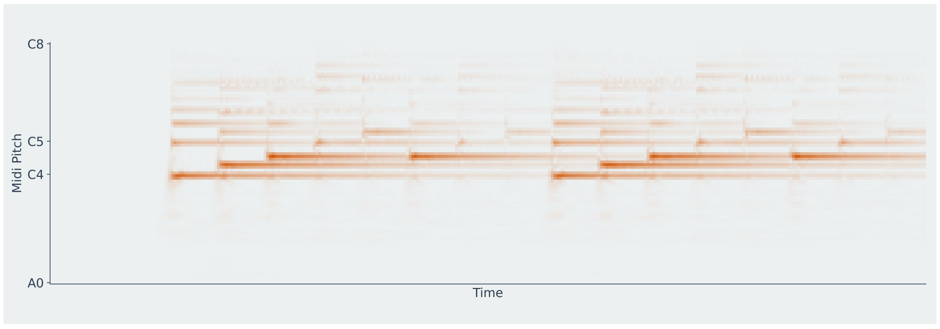
The most common approach for converting a *posteriogram* to a *piano-roll* is cropping at some value. The value 0.5 is naturally chosen for neural-network models. Ycart and Benetos, 2018 shows that *LSTM* can be trained on the posteriogram to improve the resulting piano-roll representation. This work investigates the its results and trains more elaborate *LSTM* network.



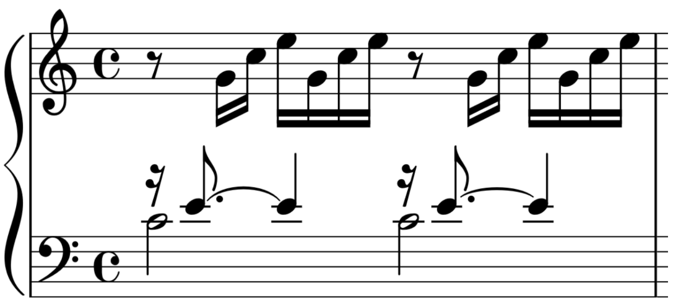
(A) *Posteriogram* used in Ycart and Benetos, 2018



(B) Corresponding *piano-roll* representation



(C) *CQT* visualization



(D) Corresponding sheet music

FIGURE 3.1: Analyzing an extract from J.S. Bach: Prelude and Fugue in C major, Well Tempered Clavier, book 1

Chapter 4

Model

This section describes a model used for converting a *posteriogram* into the *piano-roll* representation on a frame level.

4.1 Classification Task

The problem outlined in the previous chapter is an example of a *multi-label classification* task. For each frame x in the *posteriogram*, a corresponding *piano-roll* vector $\hat{y}$ should be predicted. Because several notes can be played at the time, $\hat{y}$ can belong to multiple classes simultaneously (there are 88 classes, each of which is responsible for a particular note). It is worth mentioning that the problem cannot be treated as a one-class classification task because the possible number of piano keys combinations (2^{88}) is tremendously high.

4.2 Model Architecture

Same as in Ycart and Benetos, 2018, *Long Short Term Memory (LTSM)* network is be used to solve this task (Hochreiter and Schmidhuber, 1997). It is a variation of *Recurrent Neural Networks (RNN)* which have proven themselves to perform well on variable-length input data. This is achieved by combining an input vector with a state vector to produce a new state vector that will be used for the next input.

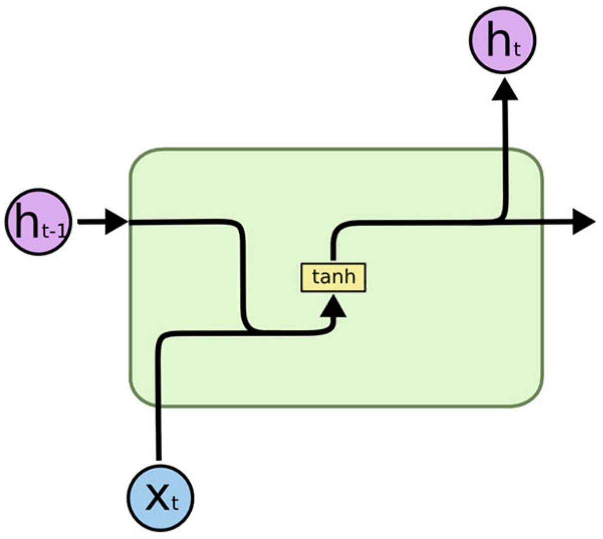


FIGURE 4.1: *RNN* cell. Prediction for time stamp t is based on previous hidden state h_{t-1} and current input x_t

Because basic *RNN* was bad at capturing long-term dependencies within the data, LSTM network was proposed. Its peculiarity is the additional cell state, which is responsible for the long term dependencies. It can be updated or left unchanged by "forget" and "update" gates and a prediction is composed of the cell state, hidden state and current input vector.

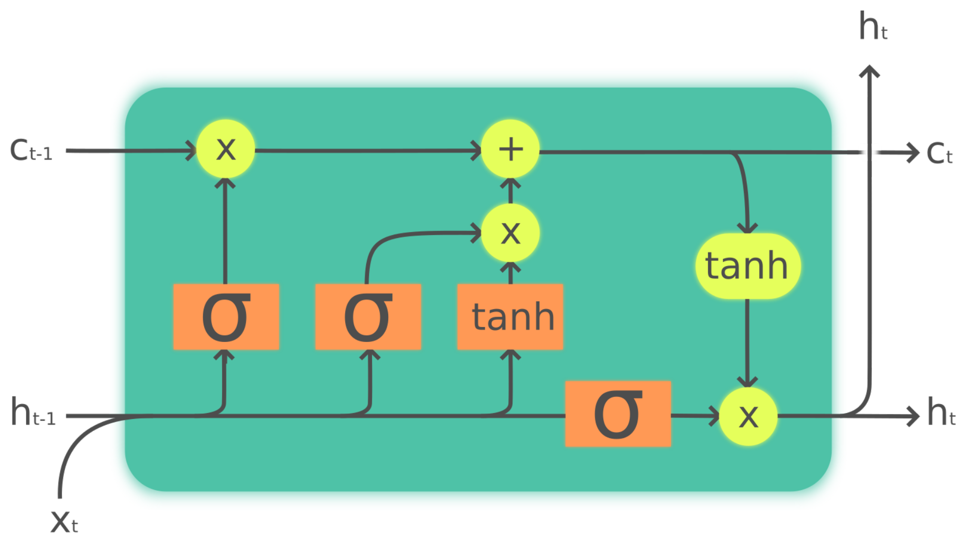


FIGURE 4.2: *LSTM* cell. Prediction for time stamp t is based on previous hidden state h_{t-1}, previous cell state c_{t-1} and current input x_t. Cell state is changed via "update" and "forget" gates. The image is an adaptation from wikipedia

.

The output of the *LSTM* is then passed through several fully-connected layers with *RELU* activation function. As the last activation function, *sigmoid* is used. It transforms the activations into $[0;1]$ range, which are treated as probabilities of the notes being played during the frame.

4.3 Posteriogram Types

As an experiment, *CQT* is used as a *posteriogram* to see whether *LSTM* can learn piano harmonics directly. As the main posteriogram, results from Benetos, Weyde, et al., 2015 are used. They present a *PLCA* based approach, which is trained on the isolated notes from *MAPS* dataset (Emiya, Badeau, and David, 2009, for sample posteriogram, see 3.1a).

Chapter 5

Dataset

For model training and evaluation MAPS dataset is used (Emiya, Badeau, and David, 2009). It is de facto primary resource for *MIR* tasks evaluations. It is an annotated dataset, which provides CD-quality recordings and ground truth midi files for over 65 hours of piano recordings on various instruments and conditions. These include both high-quality software synthesised sounds and recordings of real disklavier pianos (instruments that are capable of reproducing midi files without a human pianist).

The dataset consists of 4 main parts:

- *ISOL* - isolated notes (for monophonic pitch estimation)
- *UCHO* - recordings of common Western music chords (for works that incorporate prior harmonic knowledge)
- *RAND* - recordings of random chords
- *MUS* - 238 pieces of classical music performed on different instruments

For every composition, 3 files are provided:

- *wav* file - the actual recording or synthesis
- *midi* file - original midi file, that was played/synthesised
- *txt* file - original midi file representation, that contains onset/offset and pitch information of every note event of the original file

In this work only *MUS* part of the dataset is used for training and evaluation purposes.

Chapter 6

Metrics

In order to see the model's progress and weak spots, standard multi-label classification metrics are calculated during the training process, namely *precision*, *recall* and *f-score*. All of the metrics are calculated for each frame of the dataset and mean is taken to summarize the value over the dataset/batch.

For a single frame, let's define $\hat{y} \in \{0,1\}^{88}$ to be a prediction vector and $y \in \{0,1\}^{88}$ to be a corresponding target taken from the *piano-roll* representation (88 stands for the number of piano keys).

Metrics that are not defined for some frame (i.e. no positive samples in case of *precision*, etc.) are replaced with 0.

6.1 Precision

Precision is defined as percentage of true positives out of all positive predictions:

$$precision = \frac{TP}{TP + FP}$$

where *TP* stands for *true positives* and *FP* for *false positives* respectively.

In terms of $\hat{y}$ and y it rewrites to:

$$precision = \frac{\sum_{i=1}^{88} \hat{y}_i \cdot y_i}{\sum_{i=1}^{88} \hat{y}_i}$$

In other words, *precision* indicates, how often the model is correct when it predicts the positive sample.

6.2 Recall

Recall is defined as percentage of correctly predicted positive samples out of all target positive samples:

$$recall = \frac{TP}{TP + FN}$$

where *TP* stands for *true positives* and *FN* for *false negatives* respectively.

In terms of $\hat{y}$ and y it rewrites to:

$$recall = \frac{\sum_{i=1}^{88} \hat{y}_i \cdot y_i}{\sum_{i=1}^{88} \hat{y}_i \cdot y_i + \sum_{i=1}^{88} (1 - \hat{y}_i) \cdot y_i}$$

6.3 F-score

F-score is defined as harmonic mean of *precision* and *recall*:

$$fscore = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

It is a metric that summarizes *precision* and *recall* and gives a single number that represents how the model is performing.

6.4 Accuracy

Accuracy is not being calculated because of its needlessness. It is defined as the number of correctly classified samples over the total number of samples within 1 frame:

$$accuracy = \frac{\sum_{i=1}^{88}(1 - (\hat{y}_i \oplus y_i))}{88}$$

As a tiny amount of notes compared to the number of piano keys can be played at the time due to physical constraints, even the meaningless model (i.e. the model that predicts complete silence) can have very high accuracy because of the dataset imbalance. Hence, this metric does not provide an insight into how good the model is.

Chapter 7

Experiments

7.1 Data

Posteriograms from Benetos, Weyde, et al., 2015 are available in the scope of Ycart and Benetos, 2018. For each song from the *MUS* part of *MAPS* dataset, a corresponding posteriogram matrix with 10 milliseconds resolution is provided.

As for *CQT*, *MUS* piano pieces were processed with *hanning window* with hop length of 512, a minimal frequency that corresponds to *"A0"* pitch and 88 bins, which resulted in ≈ 11 milliseconds resolution.

Piano-roll representation was extracted from the corresponding midi files from *MAPS* dataset. Posteriogram matrix resolution was taken into account.

Because some classical pieces are duplicated (played on different instruments in different conditions) train/test split cannot be performed by shuffling the dataset. For this purpose, configuration 1 from Sigtia, Benetos, and Dixon, 2016 is used. It provides four folds of train/test split, which ensures that no piece is in train and test datasets simultaneously. For training and evaluation, the first fold is used.

Before training, all the pieces are sliced into 1-minute extracts and shuffled on every epoch of the training.

7.2 Training

The model is optimized via gradient descent using *Adam* optimizer with adaptive learning rate. Initial rate is set to 1×10^{-3} with 0.99 decay after every five epochs. *LSTM* hidden state size, number of fully connected layers and dropout vary in different experiments. Because the dataset is extremely unbalanced, class weights are introduced in every experiment.

7.3 Loss Function

As a loss function, *binary-cross-entropy (BCE)* and *focal* are used in different experiments.

For a particular frame, let's denote $\hat{y} \in [0;1]^{88}$ to be a vector of real numbers that represent probabilities of the frame belonging to each class and $y \in \{0,1\}^{88}$ to be a target vector. Then binary cross-entropy loss will be calculated as follows (see 7.1a):

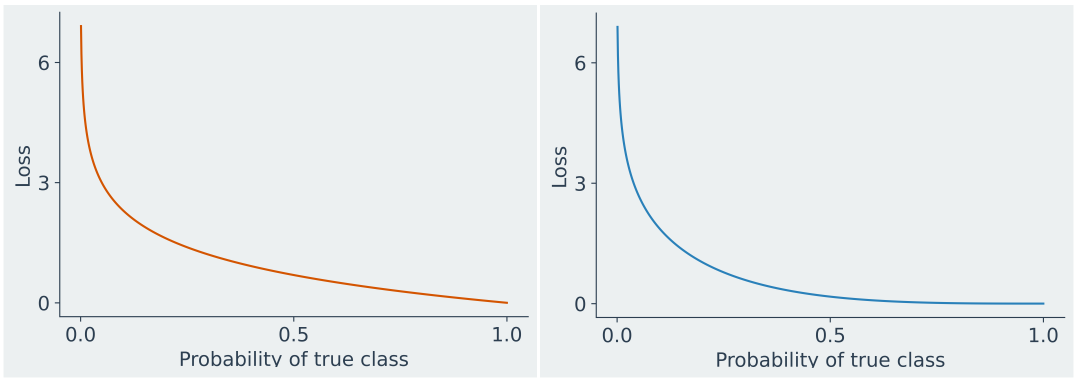
$$BCE(y,\hat{y}) = -\frac{1}{88}\sum_{i=1}^{88}(y_i \cdot \log(\hat{y}_i) + (1-y_i)\cdot \log(1-\hat{y}_i))$$

Because the dataset is imbalanced, well classified samples (with $\hat{y} > 0.6$) might overcontribute to loss function, shadowing the samples, that are hard to predict.

In order to tackle this issue, focal loss was proposed (Lin et al., 2017, see 7.1b). It complements *BCE* with the additional multiplier (*FL* stands for *Focal Loss*):

$$FL(y, \hat{y}, \gamma) = -\frac{1}{88}\sum_{i=1}^{88}((1-\hat{y}_i)^{\gamma} \cdot y_i \cdot \log(\hat{y}_i) + \hat{y}_i^{\gamma} \cdot (1-y_i) \cdot \log(1-\hat{y}_i))$$

By doing so, it flattens the original *BCE* curve (see 7.1) by emphasizing poorly predicted samples. In fact, *BCE* corresponds to *FL* with $\gamma = 0$.



(A) *Binary Cross-Entropy* loss (B) Focal loss with $\gamma = 2$

FIGURE 7.1: *BCE* and focal loss comparison.

7.4 Experiments

Experiments parameters are shown in the following table:

Experiment Name	Posteriogram Type	LSTM configuration	Dropout	Fully-Connected Layers	Loss Function
Experiment 1	CQT	hidden state size: 176 layers: 2 bidirectional: true	0.0	352 ->176 ->88	BCE
Experiment 2	PLCA	hidden state size: 176 layers: 2 bidirectional: true	0.0	352 ->176 ->88	BCE
Experiment 3	PLCA	hidden state size: 250 layers: 2 bidirectional: true	0.1	500 ->250 ->88	FL gamma = 2.0

TABLE 7.1: Experiments configurations

7.5 Results

In this section, the resulting metrics for each every experiment are presented. The next subsections provide a visualization of the model performance on a single sample from test dataset.

As a test sample, an extract from J.S.Bach: Prelude and Fugue in C major, Well Tempered Clavier, book 1 is used. See 3.1 for posteriograms, piano-roll representation and corresponding sheet music.

7.5.1 Comparison Table

Experiment Name	Precision train/test	Recall train/test	F-score train/test
Experiment 1	0.83/0.79	0.91/0.86	0.86/0.81
Experiment 2	0.84/0.8	**0.92/0.87**	0.87/0.82
Experiment 3	**0.87/0.82**	0.92/0.86	**0.89/0.83**

TABLE 7.2: Experiments best train/test framewise metrics

7.5.2 Experiment 1

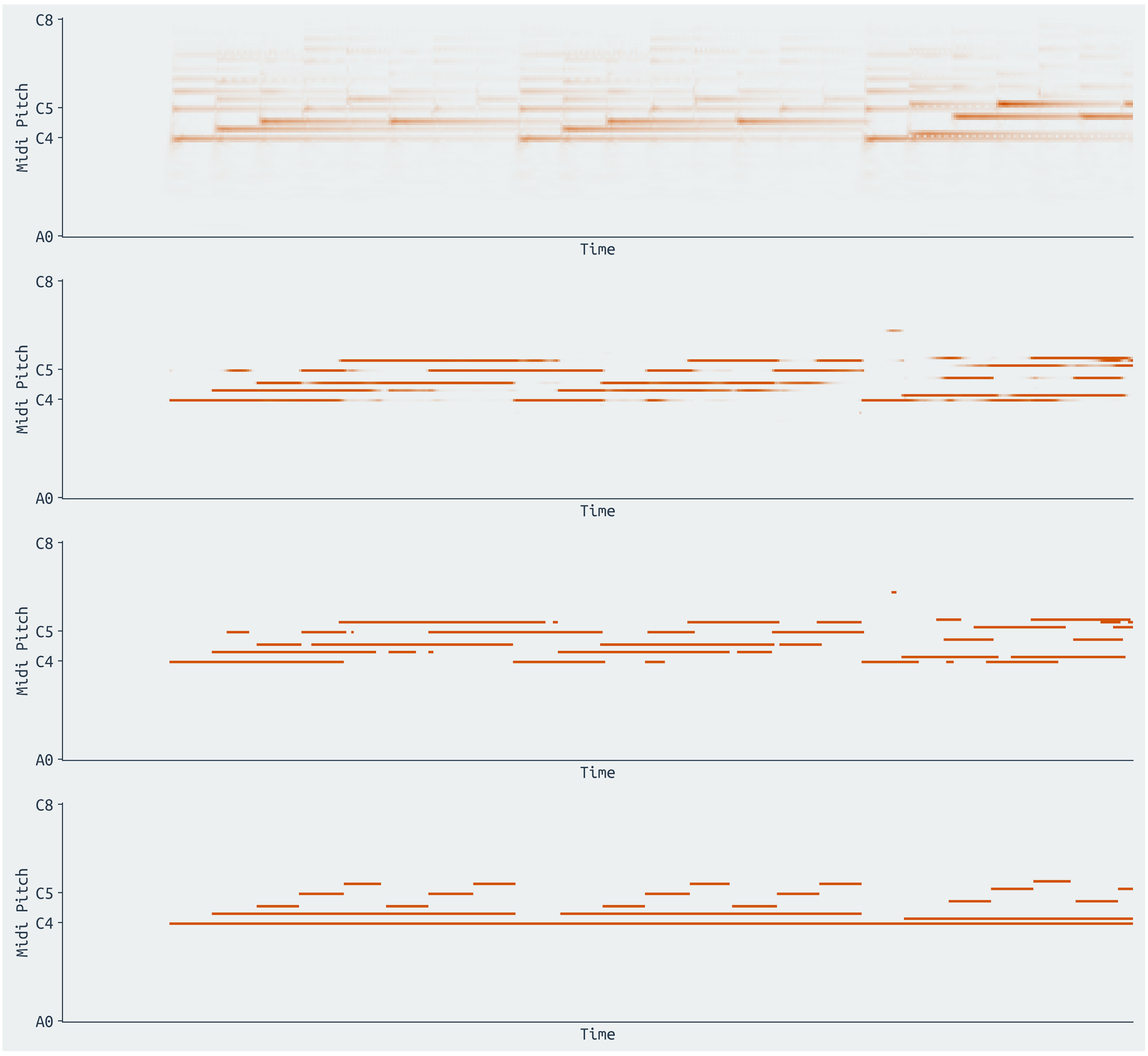


FIGURE 7.2: Experiment 1. From top to bottom: input posteriogram, LSTM activations, prediction piano-roll, actual piano-roll

7.5.3 Experiment 2

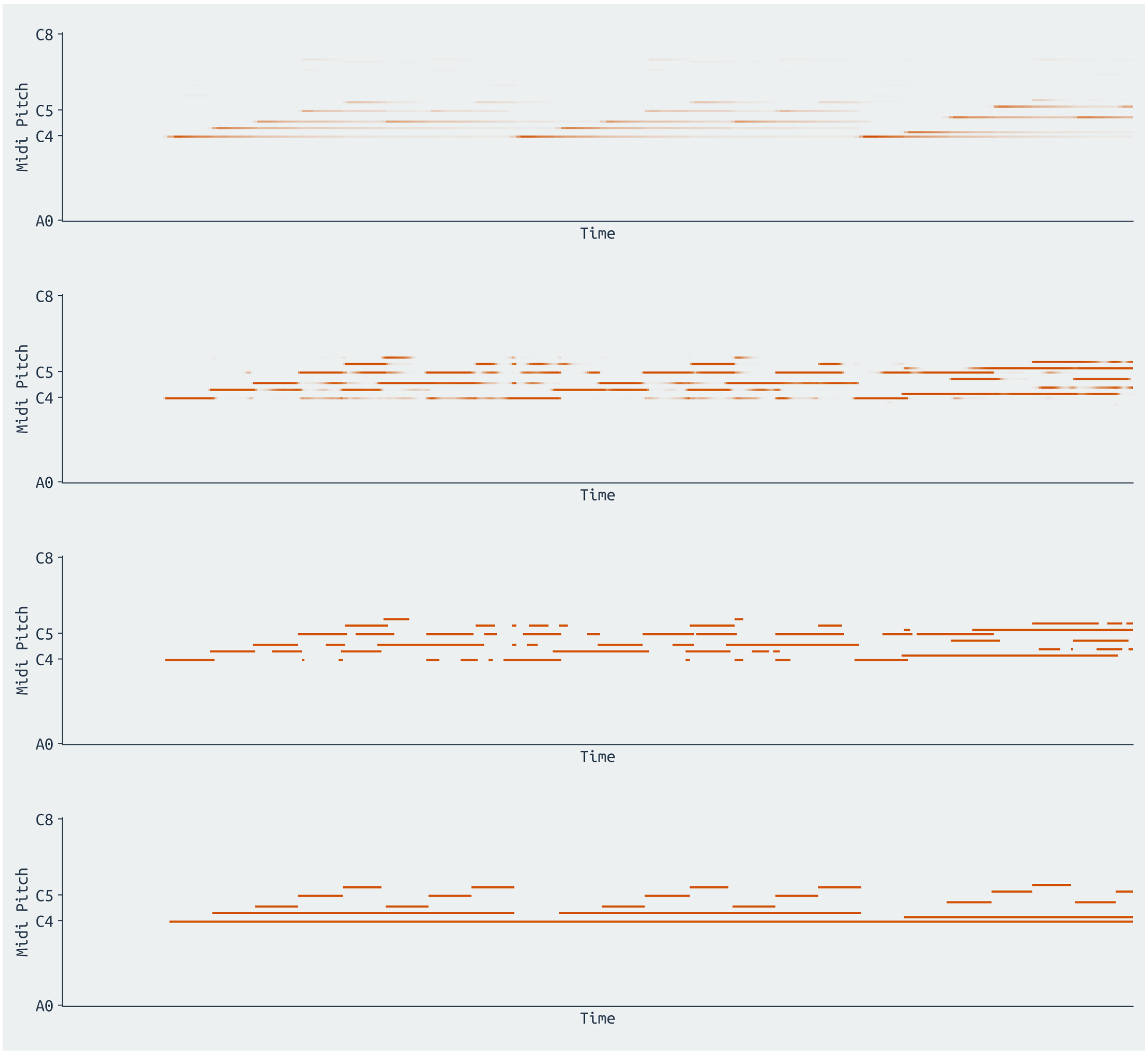


FIGURE 7.3: Experiment 2. From top to bottom: input posteriogram, LSTM activations, prediction piano-roll, actual piano-roll

7.5.4 Experiment 3

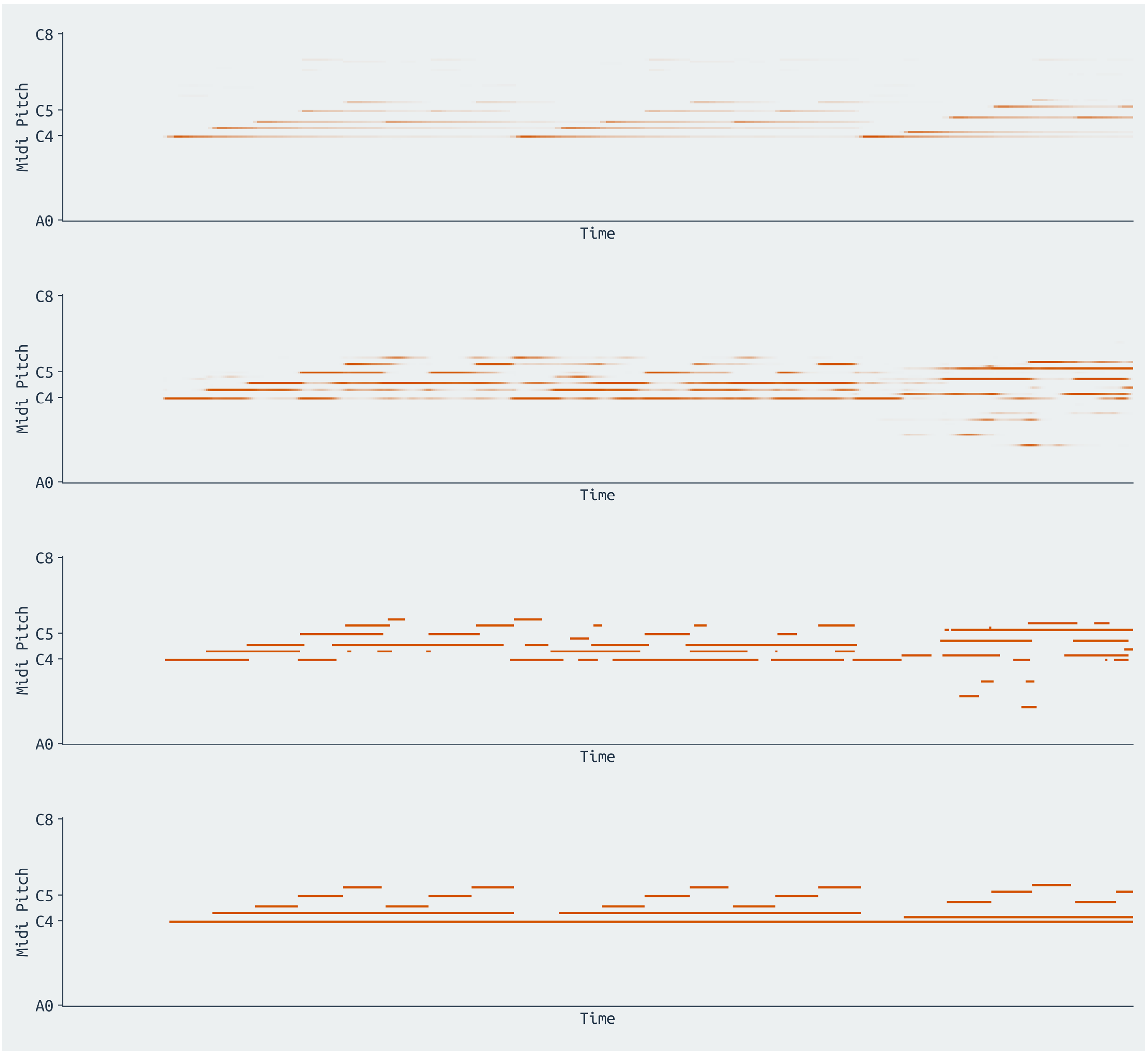


FIGURE 7.4: Experiment 3. From top to bottom: input posteriogram, LSTM activations, prediction piano-roll, actual piano-roll

Chapter 8

Results

8.1 Conclusions

8.1.1 CQT Posteriograms Results

From the first two experiments, we can see that *CQT* can be used as a posteriogram without any preprocessing (at least in current experiment setup). There are several ways to improve the posteriogram:

- downsample the signal
- try different windowing function
- incorporate *HCQT* as it is done in Bittner et al., 2017

In order to get more insights into whether this posteriogram is suitable for further experiments, note-based metrics should be calculated during training.

8.1.2 Low Recall

From the metrics curve in all the experiments we can see that test recall is dropping over the epochs:

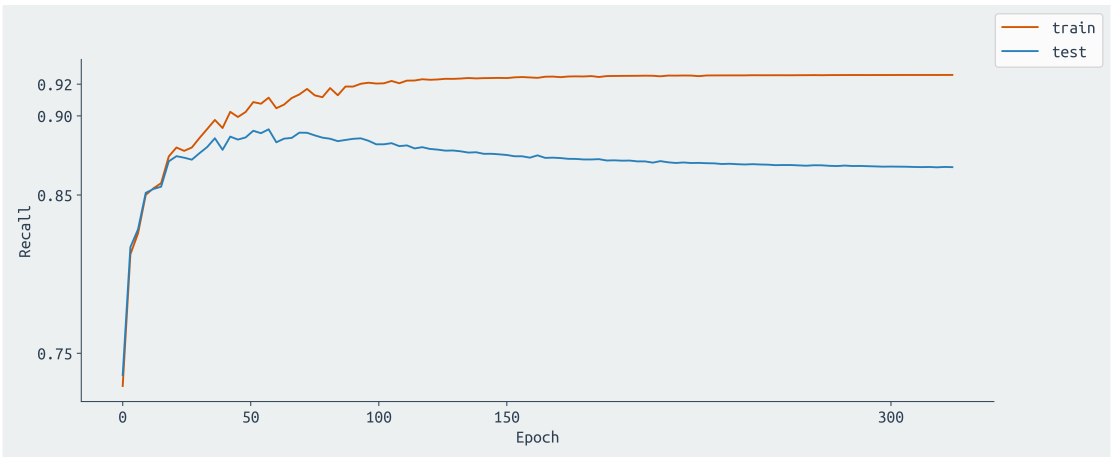


FIGURE 8.1: Recall curve for *Experiment 3*.

There might be several reasons for this behaviour:

- dataset imbalance is still a problem (*FL* with bigger γ yields similar results)
- train and test class distributions differ
- overfitting the model
- fading during sustain pedal (see 8.2.1)

8.2 Further Work

8.2.1 Fading and Sustain Pedal

One of the most obvious places for improvement is fading and sustain pedal. The piano mechanics imply that the sound arises as soon as the man presses the corresponding key. The opposite is also true: the sound goes out as soon as the key is released.

However, the second part is not true when sustain pedal comes into play. Namely, when it is on, the sound does not go off as soon as the key is released, but until the pedal is off. It means that the piano-roll representation can contain zeros, whereas the posteriogram will have non-zero coefficients at the same time.

The same is true about fading: if you press the key and hold it for some time, the sound dissolves and the reversed situation can take place: piano-roll representation will contain ones whereas the posteriogram coefficients will be close to 0.

These issues are not covered in the dataset, so some methods for fading and sustain pedal estimation should be applied.

Bibliography

Benetos, E. et al. (2019). “Automatic Music Transcription: An Overview”. In: *IEEE Signal Processing Magazine* 36.1, pp. 20–30.

Benetos, Emmanouil, Tillman Weyde, et al. (2015). “An efficient temporally-constrained probabilistic model for multiple-instrument music transcription”. In:

Bittner, Rachel M et al. (2017). “Deep Salience Representations for F0 Estimation in Polyphonic Music.” In: *ISMIR*, pp. 63–70.

Downey, Allen B. (2016). *Think DSP: Digital Signal Processing in Python*. 1st. O’Reilly Media, Inc. ISBN: 1491938455.

Emiya, Valentin, Roland Badeau, and Bertrand David (2009). “Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.6, pp. 1643–1654.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

Lin, Tsung-Yi et al. (2017). “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988.

Lins, Filipe et al. (2019). “Automatic transcription of diatonic harmonica recordings”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 256–260.

Liu, Shuchang, Li Guo, Geraint A Wiggins, et al. (2018). “A parallel fusion approach to piano music transcription based on convolutional neural network”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 391–395.

Román, Miguel A, Antonio Pertusa, and Jorge Calvo-Zaragoza (2019). “A holistic approach to polyphonic music transcription with neural networks”. In: *arXiv preprint arXiv:1910.12086*.

Román, Miguel A., Antonio Pertusa, and Jorge Calvo-Zaragoza (2019). *A holistic approach to polyphonic music transcription with neural networks*. arXiv: `1910.12086 [cs.SD]`.

Sigtia, Siddharth, Emmanouil Benetos, and Simon Dixon (2016). “An end-to-end neural network for polyphonic piano music transcription”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.5, pp. 927–939.

Smaragdis, Paris and Judith C Brown (2003). “Non-negative matrix factorization for polyphonic music transcription”. In: *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No. 03TH8684)*. IEEE, pp. 177–180.

Smaragdis, Paris, Bhiksha Raj, and Madhusudana Shashanka (2006). “A probabilistic latent variable model for acoustic modeling”. In: *Advances in models for acoustic processing, NIPS* 148, pp. 8–1.

Wu, Yu-Te, Berlin Chen, and Li Su (2019). “Polyphonic Music Transcription with Semantic Segmentation”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 166–170.

Ycart, Adrien and Emmanouil Benetos (2018). “Polyphonic music sequence transduction with meter-constrained LSTM networks”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 386–390.