

QUEEN MARY UNIVERSITY OF LONDON

SCHOOL OF ELECTRONIC ENGINEERING AND  
COMPUTER SCIENCE

---

# Music Language Models for Automatic Music Transcription

---

Adrien YCART

Submitted in partial fulfillment of the requirements of the Degree  
of Doctor of Philosophy

March 2020



centre for digital music

---

I, Adrien Ycart, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature:

Date:

Details of collaboration and publications: see Section 1.4.

## Abstract

Much like natural language, music is highly structured, with strong priors on the likelihood of note sequences. In automatic speech recognition (ASR), these priors are called *language models*, which are used in addition to acoustic models and participate greatly to the success of today’s systems. However, in Automatic Music Transcription (AMT), ASR’s musical equivalent, *Music Language Models (MLMs)* are rarely used. AMT can be defined as the process of extracting a symbolic representation from an audio signal, describing which notes were played at what time. In this thesis, we investigate the design of MLMs using recurrent neural networks (RNNs) and their use for AMT.

We first look into MLM performance on a polyphonic prediction task. We observe that using musically-relevant timesteps results in desirable MLM behaviour, which is not reflected in usual evaluation metrics. We compare our model against benchmark MLMs. We propose new intrinsic metrics to capture these aspects, and combine them into a parametric loss. We show that the loss parameters influence the behaviour of the model in consistent patterns, and that tuning them can improve AMT performance. In particular, we find no relation between MLM cross-entropy, the most commonly-used training loss, and the F-measure used in AMT evaluations.

We then investigate various methods to refine the outputs of acoustic models. First, we train neural networks to match acoustic model outputs to ground-truth binary piano rolls, a task we call *transduction*. In a first experiment, we propose the use of musically-relevant processing steps, and show that they yield better results than time-constant timesteps, although most of the improvement comes from durations being rhythmically quantised. We investigate various neural architectures and training methods for this task, and show that our proposed Convolutional Neural Network (CNN) architecture trained with the F-measure loss results in the greatest improvement.

We also use symbolic MLMs to refine the outputs of an acoustic model, by favouring output sequences deemed more likely by the MLM. We propose a novel blending model to dynamically combine the predictions from the acoustic model and MLM. We compare the use of beat-related timesteps against timesteps of a fixed duration, showing that using beat-related timesteps improves results, even when using noisy, automatically-detected beat positions.

Finally, we investigate the perceptual relevance of common AMT metrics. We conduct an online listening test to assess the similarity between benchmark AMT system outputs and the original input. We look into the agreement between ratings and AMT metrics, showing that while agreement is high in clear-cut cases, it drops when the difference in metrics is lower. We propose a new evaluation metric trained to approximate ratings based on newly-proposed musical features, showing significantly better agreement with ratings than previous metrics.

# Contents

|   |           |
|---|-----------|
| <b>List of Figures</b>                            | <b>11</b> |
| <b>List of Tables</b>                             | <b>13</b> |
| <b>List of Abbreviations</b>                      | <b>14</b> |
| <b>Acknowledgements</b>                           | <b>15</b> |
| <b>1 Introduction</b>                             | <b>17</b> |
| 1.1 Overview . . . . .                            | 17        |
| 1.2 Contributions . . . . .                       | 20        |
| 1.2.1 Intrinsic MLM evaluation . . . . .          | 20        |
| 1.2.2 Extrinsic MLM evaluation . . . . .          | 21        |
| 1.2.3 AMT evaluation . . . . .                    | 22        |
| 1.2.4 Datasets . . . . .                          | 23        |
| 1.3 Thesis outline . . . . .                      | 23        |
| 1.4 Publications . . . . .                        | 24        |
| 1.5 Research visits . . . . .                     | 25        |
| 1.6 Invited Talks . . . . .                       | 25        |
| <b>2 Background</b>                               | <b>27</b> |
| 2.1 Introduction . . . . .                        | 27        |
| 2.2 Music representations . . . . .               | 27        |
| 2.2.1 Sheet music representations . . . . .       | 27        |
| 2.2.2 Symbolic music representations . . . . .    | 28        |
| 2.2.3 Audio representations . . . . .             | 30        |
| 2.3 Automatic Music Transcription . . . . .       | 31        |
| 2.3.1 Definition . . . . .                        | 31        |
| 2.3.2 Acoustic models . . . . .                   | 34        |
| 2.3.3 Music-agnostic post-processing . . . . .    | 37        |
| 2.3.4 Introducing Music Language Models . . . . . | 38        |

|          |   |           |
|----------|---|-----------|
| 2.4      | Symbolic music modelling . . . . .                            | 40        |
| 2.4.1    | Monophonic vs Polyphonic music modelling . . . . .            | 40        |
| 2.4.2    | Polyphonic music sequence prediction . . . . .                | 42        |
| 2.4.3    | Other symbolic music models . . . . .                         | 43        |
| 2.5      | Music Language Models for Automatic Music Transcription . . . | 45        |
| 2.5.1    | Probabilistic models . . . . .                                | 45        |
| 2.5.2    | Neural MLMs . . . . .   | 46        |
| 2.5.3    | Limitations . . . . .   | 47        |
| 2.6      | Evaluation . . . . .  | 49        |
| 2.6.1    | Evaluation of MLMs . . . . .                                  | 49        |
| 2.6.2    | Benchmark AMT evaluation metrics . . . . .                    | 50        |
| 2.6.3    | Limitations and efforts for better AMT metrics . . . . .      | 51        |
| 2.7      | Available datasets . . . . .                                  | 52        |
| 2.7.1    | Symbolic music . . . . .                                      | 53        |
| 2.7.2    | Aligned MIDI and Audio . . . . .                              | 54        |
| 2.8      | Deep Learning for Sequential and Symbolic Data . . . . .      | 56        |
| 2.8.1    | HMMs and RNNs . . . . .                                       | 56        |
| 2.8.2    | The vanishing/exploding gradient problem . . . . .            | 58        |
| 2.8.3    | Long Short-Term Memory units . . . . .                        | 58        |
| 2.8.4    | Convolutional Neural Networks . . . . .                       | 60        |
| 2.8.5    | Binary neurons . . . . .                                      | 61        |
| <b>3</b> | <b>Polyphonic Music Sequence Prediction</b>                   | <b>63</b> |
| 3.1      | Introduction . . . . .  | 63        |
| 3.2      | Experimental setup . . . . .                                  | 64        |
| 3.2.1    | Problem statement . . . . .                                   | 64        |
| 3.2.2    | Model . . . . .   | 65        |
| 3.2.3    | Dataset . . . . .   | 66        |
| 3.3      | Preliminary experiment . . . . .                              | 68        |
| 3.3.1    | Description . . . . .   | 68        |
| 3.3.2    | Quantitative analysis . . . . .                               | 68        |
| 3.3.3    | Qualitative analysis . . . . .                                | 69        |
| 3.4      | Evaluation metrics . . . . .                                  | 72        |
| 3.4.1    | Benchmark evaluation metrics . . . . .                        | 72        |
| 3.4.2    | Proposed evaluation metrics . . . . .                         | 72        |
| 3.4.3    | Discussion on the proposed evaluation metrics . . . . .       | 75        |
| 3.4.4    | Combining metrics into one loss . . . . .                     | 77        |
| 3.5      | Experiments . . . . .   | 77        |
| 3.5.1    | Influence of the time step . . . . .                          | 78        |
| 3.5.2    | Comparison with other models . . . . .                        | 81        |

---

|          |  |            |
|----------|--|------------|
| 3.5.3    | Influence of $\Theta$ with $\mathcal{S}_\Theta$ loss . . . . . | 85         |
| 3.5.4    | Application to AMT . . . . .                                   | 87         |
| 3.6      | Conclusions . . . . .  | 89         |
| <b>4</b> | <b>Polyphonic Music Sequence Transduction</b>                  | <b>92</b>  |
| 4.1      | Introduction . . . . .   | 92         |
| 4.2      | Investigating musically-relevant timesteps . . . . .           | 93         |
| 4.2.1    | Motivation and objectives . . . . .                            | 93         |
| 4.2.2    | Models . . . . .   | 94         |
| 4.2.3    | Evaluation metrics . . . . .                                   | 95         |
| 4.2.4    | Experiments . . . . .  | 97         |
| 4.2.5    | Discussion . . . . .   | 99         |
| 4.3      | Comparing various architectures . . . . .                      | 101        |
| 4.3.1    | Motivation and Objectives . . . . .                            | 101        |
| 4.3.2    | Models . . . . .   | 102        |
| 4.3.3    | Training objectives . . . . .                                  | 104        |
| 4.3.4    | Evaluation metrics . . . . .                                   | 105        |
| 4.3.5    | Experiments . . . . .  | 106        |
| 4.3.6    | Comparison with time-based timesteps . . . . .                 | 109        |
| 4.3.7    | Discussion . . . . .   | 111        |
| 4.4      | Conclusion and perspectives . . . . .                          | 112        |
| <b>5</b> | <b>Music Language Model Decoding</b>                           | <b>114</b> |
| 5.1      | Introduction . . . . .   | 114        |
| 5.2      | Proposed System . . . . .                                      | 115        |
| 5.2.1    | Acoustic Model . . . . .                                       | 116        |
| 5.2.2    | Language Model . . . . .                                       | 116        |
| 5.2.3    | Blending Model . . . . .                                       | 117        |
| 5.2.4    | Search Process . . . . .                                       | 119        |
| 5.3      | Experimental setup . . . . .                                   | 119        |
| 5.3.1    | Data . . . . .   | 119        |
| 5.3.2    | Timesteps . . . . .  | 120        |
| 5.3.3    | Configurations . . . . .                                       | 121        |
| 5.3.4    | Metrics . . . . .  | 121        |
| 5.3.5    | Baselines . . . . .  | 122        |
| 5.3.6    | Training . . . . .   | 123        |
| 5.4      | Results . . . . .  | 124        |
| 5.4.1    | Comparison with baseline . . . . .                             | 124        |
| 5.4.2    | Timestep . . . . .   | 125        |
| 5.4.3    | Blending model . . . . .                                       | 127        |

|          |  |            |
|----------|--|------------|
| 5.4.4    | Scheduled sampling . . . . .                               | 127        |
| 5.4.5    | Onset-offset Evaluation . . . . .                          | 132        |
| 5.4.6    | Qualitative comparison . . . . .                           | 132        |
| 5.5      | Conclusion and perspectives . . . . .                      | 134        |
| <b>6</b> | <b>Evaluation of Automatic Music Transcription Systems</b> | <b>136</b> |
| 6.1      | Introduction . . . . .                                     | 136        |
| 6.2      | Study design . . . . .                                     | 137        |
| 6.2.1    | Stimulus design . . . . .                                  | 137        |
| 6.2.2    | User data . . . . .  | 138        |
| 6.2.3    | Setup . . . . .  | 139        |
| 6.2.4    | Participants . . . . .                                     | 141        |
| 6.3      | Results . . . . .  | 141        |
| 6.3.1    | Benchmark system performance . . . . .                     | 141        |
| 6.3.2    | Perceptual ranking of systems . . . . .                    | 142        |
| 6.3.3    | Agreement between ratings and benchmark metrics . . . . .  | 142        |
| 6.3.4    | Reported difficulty . . . . .                              | 146        |
| 6.3.5    | Analysis of confident answers . . . . .                    | 148        |
| 6.3.6    | Inter-rater agreement . . . . .                            | 149        |
| 6.3.7    | Discussion . . . . .                                       | 151        |
| 6.4      | Defining a new metric . . . . .                            | 152        |
| 6.4.1    | Comments from participants . . . . .                       | 152        |
| 6.4.2    | Feature description . . . . .                              | 152        |
| 6.4.3    | Model fitting . . . . .                                    | 155        |
| 6.4.4    | Experiments . . . . .                                      | 156        |
| 6.5      | Discussion . . . . .                                       | 159        |
| 6.6      | Reproducibility . . . . .                                  | 162        |
| <b>7</b> | <b>Conclusions and Perspectives</b>                        | <b>163</b> |
| 7.1      | Summary . . . . .  | 163        |
| 7.2      | Limitations and Perspectives . . . . .                     | 165        |
| 7.2.1    | Stylistic aspects . . . . .                                | 165        |
| 7.2.2    | Note-based MLMs . . . . .                                  | 165        |
| 7.2.3    | Self-similarity . . . . .                                  | 166        |
| 7.2.4    | Higher-level musical rules . . . . .                       | 167        |
| 7.2.5    | MLMs for Complete Music Transcription . . . . .            | 168        |
| 7.2.6    | Transduction vs. end-to-end training . . . . .             | 169        |
| 7.2.7    | On the usefulness of MLMs for AMT . . . . .                | 170        |
| 7.2.8    | Application-specific AMT evaluation . . . . .              | 171        |
| 7.3      | Conclusion . . . . .                                       | 172        |

---

|          |  |            |
|----------|--|------------|
| <b>A</b> | <b>A-MAPS: Augmented MAPS Dataset with Rhythm and Key Annotations</b>  | <b>175</b> |
| A.1      | Method . . . . .   | 176        |
| A.2      | Contents . . . . .   | 177        |
| A.3      | Applications . . . . .   | 178        |
| <b>B</b> | <b>Musical Features for Automatic Music Transcription Evaluation</b>   | <b>179</b> |
| B.1      | Introduction . . . . .   | 179        |
| B.2      | Data format . . . . .  | 179        |
| B.3      | Features . . . . .   | 180        |
| B.3.1    | Benchmark Framewise Precision, Recall, F-measure . . .                 | 180        |
| B.3.2    | Benchmark Onset-only Notewise Precision, Recall, F-measure             | 180        |
| B.3.3    | Benchmark Onset-offset Notewise Precision, Recall, F-measure . . . . . | 181        |
| B.3.4    | Number of mistakes in highest and lowest voice . . . . .               | 181        |
| B.3.5    | Loudness of false negatives . . . . .                                  | 183        |
| B.3.6    | Out-of-key false positives . . . . .                                   | 185        |
| B.3.7    | Specific Pitch Errors . . . . .  | 186        |
| B.3.8    | Repeated and merged notes . . . . .                                    | 187        |
| B.3.9    | Rhythm features . . . . .  | 188        |
| B.3.10   | Consonance measures . . . . .  | 191        |
| B.3.11   | Polyphony level . . . . .  | 192        |
| B.4      | Summary . . . . .  | 192        |

# List of Figures

|      |   |    |
|------|---|----|
| 2.1  | An example music score. . . . .   | 32 |
| 2.2  | An example MIDI file. . . . .   | 32 |
| 2.3  | An example list of notes, representing the same note sequence as Figure 2.1. . . . .  | 33 |
| 2.4  | An example piano roll, with a timestep of 40ms, representing the same note sequence as Figure 2.1. . . . .  | 33 |
| 2.5  | An example waveform, representing the same note sequence as Figure 2.1 played on a piano. . . . .   | 33 |
| 2.6  | An example spectrogram, representing the same note sequence as Figure 2.1 played on a piano. . . . .  | 34 |
| 2.7  | An example posteriogram, estimated piano roll, and ground truth piano roll. . . . .   | 35 |
| 2.8  | Typical AMT workflow. . . . .   | 36 |
| 2.9  | A schema of the HMM smoothing method described in [Poliner and Ellis, 2006]. . . . .  | 38 |
| 2.10 | Overview of the whole transcription process, using a transduction post-processing step. . . . .   | 40 |
| 2.11 | An example workflow using framewise MLM decoding. . . . .   | 41 |
| 2.12 | Schema of an HMM of order 1. . . . .  | 57 |
| 2.13 | Schema of an RNN. . . . .   | 57 |
| 2.14 | Schema of an LSTM unit. . . . .   | 59 |
| 3.1  | Single-layer LSTM network architecture. . . . .   | 66 |
| 3.2  | Preliminary experiment: Comparison of sigmoid outputs for various timesteps for the MIDI file <code>chpn-p7.mid</code> transposed into C. . . . . | 71 |
| 3.3  | Comparison of the prediction performance for the LSTM and baseline models for various timesteps assessed with the proposed metrics. . . . .       | 79 |

|     |  |     |
|-----|--|-----|
| 3.4 | Comparison of the prediction performance of various models for various timesteps assessed with the benchmark and proposed metrics. . . . .   | 82  |
| 3.5 | Comparison of sigmoid outputs for various models for the MIDI file <code>chpn-p7.mid</code> transposed into $C$ , with <i>note-long</i> timesteps. . .                                   | 83  |
| 3.6 | Evaluation of various MLMs trained with $\mathcal{S}_\Theta$ loss, with various $\Theta$ configurations, all with <i>note-long</i> timesteps. . . . .                                    | 85  |
| 3.7 | Comparison of sigmoid outputs for various training losses, all with time-short timesteps for the MIDI file <code>elise.mid</code> . . . . .  | 87  |
| 3.8 | AMT performance of various MLMs trained with the $\mathcal{S}_\Theta$ loss, with various $\Theta$ configurations. . . . .  | 89  |
| 4.1 | An example of input posteriogram and thresholded output of the transduction LSTM, compared to the ground truth. . . . .  | 96  |
| 4.2 | Architecture of the proposed CNN transduction model. . . . .   | 103 |
| 4.3 | Model comparison across training objectives, transduction and acoustic models. . . . .   | 107 |
| 5.1 | The proposed system for MLM decoding. . . . .  | 116 |
| 5.2 | An example where EST-beat steps perform better than GT-beat steps. EST-beat steps are longer, which helps smooth out spurious notes. . . . .   | 128 |
| 5.3 | An example where EST-beat steps perform better than GT-beat steps. EST-beat steps are shorter, which allows transcribing the short notes. . . . .  | 129 |
| 5.4 | An example where EST-beat steps perform worse than GT-beat steps. EST-beat steps are too long, so shorter notes are not properly represented. . . . .                                    | 130 |
| 5.5 | An example where EST-beat steps perform worse than GT-beat steps. EST-beat steps are too short, which encourages spurious notes. . . . .   | 131 |
| 5.6 | Increase of $F_{n,On}$ of PM+S over PM plotted against $F_{n,On}$ of Kelz for each piece, with 16th-note timesteps. . . . .  | 133 |
| 5.7 | Increase of $F_{n,On}$ of PM+S over PM plotted against $F_{n,On}$ of Kelz for each piece, with EST-beat timesteps. . . . .   | 133 |
| 6.1 | Screenshot of the listening test website. . . . .  | 140 |
| 6.2 | Vote proportion in pairwise comparisons of the systems. . . . .  | 143 |
| 6.3 | Percentage of agreement, across all examples, between raters and various evaluation metrics ( $F_f$ with various frame sizes, and $F_{n,On}$ with various tolerance thresholds). . . . . | 144 |

|      |  |     |
|------|--|-----|
| 6.4  | Percentage of agreement, across all examples, between raters and $F_{n,OnOff}$ , with various onset and offset tolerance thresholds. . . . | 144 |
| 6.5  | Percentage of agreement, for each pair of systems, between ratings and benchmark evaluation metrics. . . . .                               | 145 |
| 6.6  | Agreement between ratings and $F_{n,On}$ for each reported difficulty level. . . . .   | 146 |
| 6.7  | Proportion of difficulty ratings for each pair of systems. . . . .   | 147 |
| 6.8  | Distribution of difference in $F_{n,On}$ between the two options for each reported difficulty level. . . . .                               | 148 |
| 6.9  | Percentage of agreement depending on the difference in $F_{n,On}$ between the two options, computed on confident answers only. .           | 150 |
| 6.10 | $A_{conf}$ measure for each tested configuration, averaged across folds.   | 158 |
| B.1  | Decay rates per note, for various velocities (taken from [Cheng, 2016]). . . . .   | 184 |

# List of Tables

|     |   |     |
|-----|---|-----|
| 2.1 | Summary of the main symbolic music datasets. . . . .  | 54  |
| 2.2 | Summary of the main AMT datasets. . . . .   | 56  |
| 3.1 | Prediction performance for various timesteps assessed with the benchmark metrics (defined in Section 3.4.1). . . . .                                  | 69  |
| 3.2 | Prediction performance for various timesteps assessed with the proposed metrics. . . . .  | 78  |
| 3.3 | Comparison of various MLMs, for note-short and note-long timesteps. . . . .   | 84  |
| 3.4 | Comparison of AMT performance for various MLMs. . . . .   | 90  |
| 4.1 | Benchmark evaluation metrics for all systems, evaluated on the MAPS subsets ENSTDkCl and ENSTDkAm. . . . .  | 98  |
| 4.2 | Comparison of transduction results using note-based processing, and time-based processing with post-quantisation of the outputs. . . . .              | 100 |
| 4.3 | Transduction results as percentages for median models with Kelz input. . . . .  | 106 |
| 4.4 | Transduction results as percentages for median models with Bittner input. . . . .   | 106 |
| 4.5 | Comparison of time-based and note-based timesteps for the F-measure-trained CNN, all evaluated with time-based timesteps, with Kelz input. . . . .    | 111 |
| 4.6 | Comparison of time-based and note-based timesteps for the F-measure-trained CNN, all evaluated with time-based timesteps, with Bittner input. . . . . | 111 |
| 5.1 | Definition of the features used for the blending model. . . . .   | 118 |
| 5.2 | Comparison of MLE-trained and BO-trained HMMs, with all timesteps. . . . .  | 123 |
| 5.3 | Results of all MLM decoding experiments, with all timesteps. . . . .  | 125 |

|     |  |     |
|-----|--|-----|
| 5.4 | MLM decoding results using the onset-offset notewise metrics, using the best-performing configuration for each timestep. . . . .   | 134 |
| 6.1 | Benchmark evaluation metrics for all systems, evaluated on the MAPS subsets ENSTDkCl and ENSTDkAm. . . . .   | 142 |
| 6.2 | Coefficients and p-values for the linear mixed effects model using agreement with $F_{n,On}$ as dependent variable and features as fixed effects. . . . .                            | 145 |
| 6.3 | Coefficients and p-values for the linear mixed effects model using difficulty as dependent variable and features as fixed effects. . . .   | 148 |
| 6.4 | Coefficients and p-values for the linear mixed effects model using agreement with $F_{n,On}$ as dependent variable and features as fixed effects, on confident answers only. . . . . | 149 |
| 6.5 | Coefficients and p-values for the linear mixed effects model using agreement among raters as dependent variable and features as fixed effects. . . . .                               | 151 |
| 6.6 | Description of each tested feature configuration. . . . .  | 160 |
| A.1 | Evaluation of the beat annotations: minimum, median and maximum values in MIDI ticks of $\delta_{mean}$ and $\delta_{max}$ across the dataset  | 177 |
| B.1 | Rhythmic feature means and standard deviation (std) across all stimuli, with 4 levels of rhythmic precision. . . . .   | 191 |
| B.2 | Summary of all the proposed evaluation metrics. . . . .  | 194 |

# List of abbreviations

## Abbreviations

|      |                                     |
|------|-------------------------------------|
| AMT  | Automatic Music Transcription       |
| ASR  | Automatic Speech Recognition        |
| HMM  | Hidden Markov Model                 |
| LSTM | Long Short-Term Memory              |
| MLM  | Music Language Model                |
| NLP  | Natural Language Processing         |
| NMF  | Non-negative Matrix Factorisation   |
| PLCA | Principal Latent Component Analysis |
| RBM  | Restricted Boltzmann Machine        |
| RNN  | Recurrent Neural Network            |

## Notations

|               |                                 |
|---------------|---------------------------------|
| $M$           | Piano roll                      |
| $M_t$         | $t$ -th frame of a piano roll   |
| $\hat{M}$     | Estimated piano roll            |
| $N_p$         | Number of pitches considered    |
| $P$           | Posteriogram                    |
| $T$           | Number of timesteps             |
| $(s, e, p)$   | Music note                      |
| $s$           | Music note onset (start) time   |
| $e$           | Music note offset (end) time    |
| $p$           | Music note pitch                |
| $\mathcal{H}$ | Cross-entropy                   |
| $P_f$         | Framewise Precision             |
| $R_f$         | Framewise Recall                |
| $F_f$         | Framewise F-Measure             |
| $P_{n,On}$    | Onset-only notewise Precision   |
| $R_{n,On}$    | Onset-only notewise Recall      |
| $F_{n,On}$    | Onset-only notewise F-Measure   |
| $P_{n,OnOff}$ | Onset-only notewise Precision   |
| $R_{n,OnOff}$ | Onset-offset notewise Recall    |
| $F_{n,OnOff}$ | Onset-offset notewise F-Measure |

# Acknowledgements

First and foremost, I would like to express my deepest gratitude to my main supervisor, Emmanouil Benetos, for his support and guidance throughout this three-year journey. I will always be impressed with how he manages to make time, not only for me, but for everyone, and still keep up with everything. His feedback has always been extremely relevant, pragmatic, and has had a paramount influence on this work, undoubtedly for the better. He has taught me a lot, not only from a strictly scientific point of view, but also in many other aspects of a researcher's life (research management and methods, publications, networking...), and for this I will always be grateful.

The completion of this dissertation would not have been possible without my secondary supervisor, Marcus T. Pearce, who has also provided great guidance on various aspects of this project. Many times, he helped put things in perspective, and set a course for this work focusing on the important things when I was getting lost in heaps of experiments. The regular meetings with him and the Music Cognition lab have also taught me a lot in domains I was not familiar with, and helped me broaden my horizons. I would also like to extend my gratitude to my independent assessor, Simon Dixon, who provided great feedback during the various assessment stages of this thesis. I also wish to express in advance my deepest appreciation to my two examiners, H el ene-Camille Crayencour and Robin Laney, for the attention they will give to my research, and the feedback they will give which, I am sure, will help improve it.

I would also like to thank the people who collaborated with me and participated in the making of this work. I am deeply indebted to Andrew McLeod, for his amazing work, for the great times we shared travelling Japan, and overall for greatly participating in making my research visit in Kyoto an amazing experience. His creativity, efficiency, and accessibility made working with him a real pleasure. I must also thank Daniel Stoller, for his great work and pugnacity in the face of sometimes disappointing results. I learned a lot at his contact, in particular in terms of software development and how to work in practice with intricate and often obscure neural networks. Thanks should also go to Lele Liu,

for taking some workload off my shoulders and greatly carrying it out.

I also wish to thank those who invited and welcomed me at various points in my PhD. I would like to express my gratitude to Haruhiko Kishi, for hosting me during 3 months at Sony CSL Tokyo, and for inviting me to discover many places I would not have been able to visit as a foreigner. Many thanks to Emmanuel Deruty as well, for being instrumental in making this internship happen. I am also extremely grateful to Kazuyoshi Yoshii, for very kindly inviting me twice, first for a day, then for a month, at Kyoto University, and making this great scientific collaboration and personal experience possible. Thanks also to Artur d'Avila Garcez, for allowing me to follow his Neural Networks class at City University at the start of my PhD. I also wish to thank Keiji Hirata, Patrick E. Savage, Masataka Goto and Masatoshi Hamanaka, who have invited me to give talks, visit their lab, and have many great discussions when I was in Japan.

I would also like to thank the many people who have taken the time to discuss my research with me, offering very valuable feedback, and helping shape the work that is presented in this thesis. With no particular order, and apologising in advance for anyone I might have forgotten, I thank Daniel Müllensiefen, Tillman Weyde, Daniel Wolff, Eita Nakamura, Peter Harrison, Rémi de Fleurian, Dan Stowell, Emir Demirel, Tian Cheng, Rainer Kelz and Siddharth Sigtia. These discussions were great fuel for my research and it would not have been what it is without them.

I would also like to gratefully acknowledge the essential support I received from Haris Krikelis and Simon Butcher in using Queen Mary's IT infrastructure, and thank the QMUL and EECS administrative staff for making this whole machinery work.

Finally, I would also like to thank all the people who have surrounded me during this period of my life. Many thanks to the people from C4DM and the Music Cognition lab, for creating this great work environment to evolve in, and for many interesting discussions, research-related or not, and many great shared moments, at lunch or at the pub. I am also very grateful to my friends in London, Paris, Grenoble, and elsewhere, for taking my mind off things, and to my family, for their unwavering support. Finally, I would like to say a special thanks to Farah El Hachami, for always being by my side before, during, and hopefully long after these three years.

# Chapter 1

## Introduction

### 1.1 Overview

Music and spoken language have many common features. Both are made of successions of sounds, and can be transcribed to a written, symbolic form, such as text and music score. Both possess sequential structure, and follow a specific set of rules, albeit fuzzy (grammar and music theory<sup>1</sup>). In both cases, using prior knowledge, either pre-specified or learned, this underlying set of rules can be exploited to fill in missing parts in sequences, to some extent, as several possibilities can often be accepted. More broadly, these rules can be used to determine what does or does not make a valid sequence. In other words, a likelihood can be assigned to sequences, although this likelihood might depend on some parameters, for instance dialect or musical style.

Such models of word sequences, known as *language models*, can be useful for a wide variety of natural language processing (NLP) applications: machine translation, spelling correction, and question answering amongst others. Similarly, *music language models* (MLMs) can be useful for various applications: they can be used for music generation, since models of symbolic music, essentially defining a probability distribution over music sequences, can be used similarly to infer or generate. Predictive models of music also have applications in fields such as computational musicology, as they can capture some stylistic aspects of music, and music cognition, by modelling music expectation.

One of the most notable uses of language models is automatic speech recognition, where they have been combined with audio feature extraction modules known as *acoustic models* for a long time, and greatly contribute to the success

---

<sup>1</sup>We use here the term “music theory” to describe theoretical tools used to describe common uses in Western music. It does not equate to a scientific theory; in particular, an observation that disagrees with some music theory rules does not invalidate them.

of today’s methods [Huang and Deng, 2010]. The musical equivalent of automatic speech recognition is automatic music transcription (AMT), that is to say, the conversion of a music audio signal into a symbolic representation describing what notes were played, and when, usually in the form of a time-pitch representation called *piano roll*, or a list of note events characterised by their pitch, onset time and duration, similar to a MIDI file [Benetos et al., 2019]. Transcribing music to a complete staff-notation music score, which we call in this dissertation Complete Music Transcription (CMT), requires other steps (beat tracking, meter and key estimation, rhythm transcription, pitch spelling...). We leave these steps outside the scope of this dissertation, and focus on descriptive transcription (*i.e.* what exactly was played by the performer) rather than prescriptive transcription (*i.e.* how it would have to be written for the performance to be reproduced). AMT is useful in a variety of contexts, such as music education (automatic tutoring of instrument practice), music creation and production (automatic accompaniment), musicology (analysis of improvised music), or in further MIR tasks (cover detection). For AMT, MLMs have only been introduced fairly recently, and are not explicitly used in most state-of-the-art systems [Kelz et al., 2016, Hawthorne et al., 2018, Kelz et al., 2019].

Modelling music is a vast topic, and the methods used depend greatly on the kind of music considered. Western classical tonal music can be divided in 3 main categories: monophonic, homophonic and polyphonic. Monophonic music corresponds roughly to melodies; there is not more than one note at a given time. Homophonic music corresponds to pieces where there is a melody and an accompaniment. It can be modelled as two monophonic sequences, one for the melody notes, and one for the chord symbols. In polyphonic music, there can be an arbitrary number of notes sounding together, forming an arbitrary number of voices. There are both *vertical* dependencies (simultaneous notes forming chords) and *horizontal* dependencies (each voice forms an internally coherent melody line). This represents a big difference between music and language: spoken language is essentially monophonic, although some work has focused on multi-speaker simultaneous speech recognition [Chen et al., 2018]. We focus here on polyphonic music sequence modelling and polyphonic music transcription, which makes direct application of NLP methods difficult: in multi-speaker speech recognition, the simultaneous sentences are usually considered independent, while in music, simultaneous voices are strongly correlated.

Recurrent neural networks (RNNs) have become increasingly popular for sequence modelling in a variety of domains such as text, speech or video processing [Goodfellow et al., 2016]. In particular, long short-term memory (LSTM) networks [Hochreiter and Schmidhuber, 1997] have helped make tremendous progress in natural language modelling [Mikolov et al., 2010]. Some very com-

plex neural architectures have been designed, but as we will argue in Section 2, they often seem to be used in suboptimal ways when applied to music.

In this thesis, we explore the design of MLMs using neural networks, and their use for AMT, focusing on Western classical piano music as our application case. Generally speaking, we investigate neural models of music sequences, and methods to post-process the output of acoustic models using musical priors. Among these post-processing methods, we focus on two main approaches: *transduction methods*, where a neural network is trained in a supervised way to map outputs of acoustic models to binary piano rolls, and *MLM decoding methods*, where an MLM is used to assess the likelihood of output sequences and favour the most likely ones. In terms of methodology, we aim to root our design choices in musical knowledge, rather than directly transposing machine-learning methods to music data. In particular, our approach is to use simpler neural architectures, and study them in order to uncover their behaviour, their strengths and shortcomings.

Although we use the term MLM throughout this thesis to refer to such neural architectures, these models should not be mistaken for comprehensive models of music. Indeed, music is made of complex interactions, both in terms of vertical and horizontal dependencies, that exist on multiple time-scales, with a hierarchical structure. In particular, many of these interactions are arguably better described as symbolic, logical rules than statistical regularities. Despite showing some success in sequence modelling tasks, most of the models considered both in this thesis and in the literature cannot represent such complex interactions, in particular on long timescales. Instead, they learn some statistical patterns that can be found in music. Although this represents a simplification of what music is, we believe that this can still constitute useful biases for extrinsic tasks such as AMT. Rather than developing comprehensive models of music, we focus in this thesis on the more practical objective of investigating what kinds of patterns RNNs are able to model, to what extent they succeed, and whether these patterns can be exploited to improve AMT. More specifically, through the experiments presented here, we hope to investigate the following research questions:

**How does the input representation influence an MLM for AMT?**

This question is often overlooked in the literature (see Chapter 2), yet, it was shown in other applications to greatly impact the behaviour of MLMs [Korzeniowski and Widmer, 2017]. We investigate here specifically frame-based representations, and compare the use of small, time-constant processing steps typically used in signal processing against musically-relevant units, such as beat subdivisions. We look into their differences for a variety of models and applications (music sequence prediction in Chapter 3, music sequence transduction in

Chapter 4, AMT with MLM decoding in Chapter 5).

**Are common evaluation metrics relevant for MLMs and AMT?**

We question these benchmark metrics, investigating whether what they capture corresponds to what we expect from a good model. In particular, we aim to design metrics that can give us insight into how the model behaves on multiple dimensions rather than one single metric. We also aim to design these metrics so that they correspond as much as possible to musical concepts. Besides, we investigate, when possible, the use of these new metrics as learning objectives for our MLMs.

Aside from these general scientific questions, we also look into more specific engineering challenges:

**Which kind of neural architecture is most suitable for transduction?** We investigate various kinds of neural network architectures, output types and learning objectives for transduction in Chapter 4. We emphasise on the robustness of these findings by studying their significance on two different acoustic models, and taking into account the training process randomness.

**How can we best combine acoustic and language model predictions in MLM decoding?** We look into various methods to combine the predictions from both models. In particular, we propose in Chapter 5 to dynamically combine the two predictions using an additional blending model, enabling us to give more or less importance to each model depending on its value as well as various context features, and study the performance of this approach in various configurations.

## 1.2 Contributions

There are two main types of language model evaluation methodologies: intrinsic and extrinsic [Jurafsky and Martin, 2014]. Intrinsic evaluation assesses how well the model captures what it is supposed to model. Extrinsic evaluation assesses the usefulness of the model for a particular task. The latter is usually the one that actually matters, but the former is usually easier, because it does not require external components. Ideally, the evaluation metrics for intrinsic evaluation should correlate well with extrinsic performance measures. Besides, the extrinsic measures should correlate well with human judgement. We make contributions in the following research problems.

### 1.2.1 Intrinsic MLM evaluation

As an intrinsic evaluation task, we study polyphonic music sequence prediction. In NLP, language models are usually evaluated on their predictive power, that is

to say their ability to predict the next character or word in a sentence, through the perplexity evaluation metric [Jurafsky and Martin, 2014]. Here we consider a similar task, using as input data format piano rolls: given a piano roll, we try to predict the content of the next frame, using LSTM networks. Our contributions are the following:

- We compare various timestep configurations as input to our system, both quantitatively and qualitatively, showing that for shorter timesteps, the MLM tends to repeat the previous frame, while with musically-relevant timesteps the MLM is able to predict note transitions to some extent.
- We propose new evaluation metrics describing high-level MLM behaviour, that can be used as diagnosis tools in order to understand better what models manage or fail to do in the context of music modelling.
- We design a parametric training loss based on the new metrics, allowing us to tune the behaviour of the model for each specific application.
- We investigate the relation between loss parameters, MLM performance and AMT performance, showing in particular that while usual evaluation metrics do not correlate with AMT performance in the general case, our evaluation metrics correlate with some lower-level AMT metrics.
- We compare our model against two models found in the literature, showing that our model surpasses them (although it is also more complex in numbers of parameters).

### 1.2.2 Extrinsic MLM evaluation

As an extrinsic evaluation task, we use our models for AMT to post-process the output of an acoustic model. As mentioned in Section 1.1, we investigate two types of post-processing:

**Transduction methods**, which aim to convert one sequence of symbols into another (here, the output of an acoustic model into a binary piano roll). These methods are trained on pairs of posteriograms and aligned piano rolls, and do not necessarily generalise to unseen acoustic models.

**MLM decoding methods**, where an MLM is used to assess the likelihood of the output sequences. The MLM is trained on symbolic data, independently from the acoustic model.

On transduction, we make the following contributions:

- We compare the use of a typical sound analysis timestep of 10ms (that we call time-based) against a musically-relevant timestep of a 16th note (that we call note-based) as part of an LSTM transduction model. As a proof-of-concept experiment, we use ground-truth 16th note annotations. We show that note-based steps improve results, but only because the output is quantised, not because of a better temporal modelling by the LSTM.
- We compare various transduction models and training methods, using various acoustic model inputs, and show that our proposed convolutional neural network (CNN) architecture, using binary neurons and trained with the F-measure loss performs best.
- We compare once again this CNN architecture with time- and note-based steps, showing that the CNN allows us to improve results beyond simple quantisation only on one of the tested acoustic models.

On MLM decoding, we make the following contributions:

- We propose a novel blending model, allowing us to dynamically combine the predictions from the acoustic and language model depending on their values and some context features.
- We compare the use of time-based and musically-relevant steps, and show that musically-relevant steps improves results much further than in the case of transduction.
- We experiment using automatic beat estimations rather than ground-truth annotations, and show that surprisingly, using automatic beats gives better results on average than using ground truth beat positions.
- We show that using scheduled sampling allows us to make the MLM more robust to noise at test time, although it does not always result in an overall improvement.

### 1.2.3 AMT evaluation

On the topic of AMT evaluation, we make the following contributions:

- We run an online listening test to gather perceptual data on the quality of AMT outputs. We investigate to what extent these ratings correlate with benchmark evaluation metrics, showing only partial agreement.
- We define new musically-relevant statistics that can be computed on transcriptions. These can be used to obtain a deeper understanding of the types of mistakes made by an AMT system.

- We then define a new evaluation metric in the form of a simple logistic regression model trained on approximate ratings based on the defined statistics, showing significantly better agreement with ratings than previous evaluation metrics.
- We investigate what factors were most important to raters when judging the quality of AMT outputs by looking into the importance of these features through an ablation study.

### 1.2.4 Datasets

We release the following data:

- We release the A-MAPS dataset (see Appendix A): a new set of ground-truth MIDI files for the MIDI-aligned piano sounds (MAPS) dataset [Emiya et al., 2010], containing information about tempo, time signatures, note durations, tonality, stream separation and text annotations. These can help towards developing complete, score-format transcription systems from audio.
- We also make available the anonymised data collected in our listening test, along with the stimuli we used, to encourage further research on perceptually-relevant evaluation metrics for AMT.

## 1.3 Thesis outline

The remainder of this manuscript is organised as follows:

**Chapter 2** reviews related works on MLMs, AMT, and neural networks and discusses their limitations.

**Chapter 3** presents the experiments conducted on polyphonic music sequence prediction as an intrinsic evaluation task.

**Chapter 4** describes the various experiments we conducted on polyphonic music sequence transduction.

**Chapter 5** is focused on MLM decoding, presenting our system and its results.

**Chapter 6** investigates the perceptual relevance of evaluation metrics for AMT and describes the features and metric we propose.

**Chapter 7** discusses future directions to improve the work presented in this thesis.

## 1.4 Publications

The main publications associated with this thesis are the following:

1. Adrien Ycart and Emmanouil Benetos, “Towards a Music Language Model for Audio Analysis” *DMRN+11: Digital Music Research Network One-day Workshop*, Dec. 2016, London, United Kingdom. (Extended abstract)
2. Adrien Ycart and Emmanouil Benetos. “Neural Music Language Models: Investigating the Training Process” *10th International Conference of Students of Systematic Musicology (SysMus)*, Sept. 2017, London, United Kingdom. (Extended abstract)
3. Adrien Ycart and Emmanouil Benetos. “A Study on LSTM Networks for Polyphonic Music Sequence Modelling” *18th International Society for Music Information Retrieval Conference (ISMIR)*, Oct. 2017, Suzhou, China.
4. Adrien Ycart and Emmanouil Benetos, “Assessing the Use of Metrical Information in LSTM-based Polyphonic Music Sequence Transduction” *DMRN+12: Digital Music Research Network One-day Workshop*, Dec. 2017, London, United Kingdom. (Extended abstract)
5. Adrien Ycart and Emmanouil Benetos. “Polyphonic Music Sequence Transduction with Meter-Constrained LSTM Networks” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, Calgary, Canada.
6. Adrien Ycart and Emmanouil Benetos, “A-MAPS: Augmented MAPS Dataset with Rhythm and Key Annotations” *19th International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, Sep. 2018, Paris, France. (Extended abstract)
7. Adrien Ycart, Emmanouil Benetos, Marcus T. Pearce and Patrick E. Savage, “Towards defining a perception-based evaluation metric for automatic music transcription”, *SEMPRE Graduate Conference 2019*, Mar. 2019, Cambridge, United Kingdom. (Extended abstract)
8. Adrien Ycart, Andrew McLeod, Emmanouil Benetos and Kazuyoshi Yoshii. “Blending acoustic and language model predictions for automatic music transcription” *20th International Society for Music Information Retrieval Conference (ISMIR)*, Nov. 2019, Delft, Netherlands.
9. Adrien Ycart, Daniel Stoller and Emmanouil Benetos. “A Comparative Study of Neural Models for Polyphonic Music Sequence Transduction”

*20th International Society for Music Information Retrieval Conference (ISMIR)*, Nov. 2019, Delft, Netherlands.

10. Adrien Ycart and Emmanouil Benetos, “Learning and Evaluation Methodologies for Polyphonic Music Sequence Prediction with LSTMs” *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 28, pp. 1328-1341, 2020.
11. Adrien Ycart, Lele Liu, Emmanouil Benetos and Marcus T. Pearce, “Investigating the Perceptual Validity of Evaluation Metrics for Automatic Piano Music Transcription” *Transactions of the International Society for Music Information Retrieval*, vol. 3(1), pp. 68–81, 2020.

In (8), Andrew McLeod implemented the beam search algorithm and blending model, including the Bayesian optimisation procedure. In (9), Daniel Stoller provided his implementation of the Wasserstein Generative Adversarial Network [Arjovsky et al., 2017]. For these publications, the two first authors are stated as equal contributors to the publication. In (11), Lele Liu participated in the definition and implementation of the musical features used to define the proposed metric. In (8), Kazuyoshi Yoshii supervised and provided guidance on the experiments we conducted. In (7), Patrick Savage participated in the discussion on the design of the perceptual experiments.

In all other cases, the author was the main contributor to the publication, under the supervision of Dr. Emmanouil Benetos, and Dr. Marcus T. Pearce when indicated.

## 1.5 Research visits

During the course of this PhD, some other research labs were visited.

- From October to December 2018, I did a 3-month internship at Sony CSL Tokyo supervised by Haruhiko Kishi, working on lead sheet transcription from audio. This internship corresponded to an interruption of studies, the results of this visit are not presented in this thesis.
- In January 2019, I carried out a 1-month research visit at the Speech and Audio Processing Group of Kyoto University, and collaborated with Andrew McLeod and Kazuyoshi Yoshii.

## 1.6 Invited Talks

- “Interactive Rhythm Transcription and Music Language Models”, Invited talk at the Speech and Audio Processing Group, Kyoto University, Nov.

2017, Kyoto, Japan.

- “Music Language Models for Automatic Music Transcription ”, Invited talk at Comp Music Lab, Keio University, Nov. 2018, Fujisawa, Japan.
- “Music Language Models for Automatic Music Transcription ”, Invited talk at Music Information Intelligence Lab, RIKEN, Dec. 2018, Tokyo, Japan.
- “Music Language Models for Automatic Music Transcription ”, Invited talk at Hirata Takegawa Lab, Future University, Dec. 2018, Hakodate, Japan.
- “Music Language Models for Automatic Music Transcription ”, Invited talk at Speech and Audio Processing Lab, Kyoto University, Jan. 2019, Kyoto, Japan.
- “Music Language Models for Automatic Music Transcription ”, Invited talk at Media Interaction group, AIST, Feb. 2019, Tsukuba, Japan.

# Chapter 2

## Background

### 2.1 Introduction

In this chapter, we will introduce useful concepts for the understanding of the rest of this thesis. We will also present related models and studies from the literature. We will discuss some of their shortcomings, and will point out previous works that could be useful in the context of this thesis.

The remainder of this chapter is organised as follows. In Section 2.2, we discuss various ways to represent music. In Section 2.3, we define the problem of automatic music transcription (AMT), and present some related prior works. In Section 2.4, we present symbolic music modelling systems, in particular for polyphonic music. In Section 2.5, we discuss some works that have used musical priors to improve AMT performance. In Section 2.6, we describe how music language models and AMT systems are evaluated, and discuss potential improvements. In Section 2.7, we present some useful datasets, both for symbolic music and for AMT. Finally, in Section 2.8, we present some useful concepts about deep learning for sequential data modelling.

### 2.2 Music representations

In this subsection, we present and define various ways to represent music. In particular, we introduce notations, that we will use throughout this thesis.

#### 2.2.1 Sheet music representations

Traditionally, Western music is written as music scores, a graphical representation of music. It is the main vessel for transmitting musical ideas, in particular before music could be recorded. It can be seen like the script of a theatre play:

it contains what the performers should do, and rough indications on how they should do it, but still leaves much room to interpretation and expression. In that sense, it is a form of *prescriptive* notation, as opposed to *descriptive* notation, which represents precisely how a piece of music was performed. An example is given in Figure 2.1 (we will use the same example throughout this chapter). Although it is beyond the scope of this thesis to explain in detail how to read music (see Gould [2016] for more details on music notation), we will outline what can or cannot be described by a music score.

Music scores contain information about various aspects of a piece. The key signature gives information about its tonality, while its time signature describes its meter. The tempo at which the music should be performed is also given, either as a nominal value, as in Figure 2.1, or as textual indications (*e.g.* “Allegro”, “Largo”). It also contains prescriptive information about which pitches have to be played, and when. The *pitch* is represented by the position of the note head on the staves, while the note stem, head colour, and potential presence of flags represent the *note value*. The note value describes the duration of a note relative to the tempo and meter. We call this way of representing durations *musical time*: durations are expressed as integer fractions of the duration of a beat, which is itself defined according to the tempo. The score can also contain prescriptive performance indications, such as dynamics (how loud notes should be played), phrasing (whether to link or separate notes), or potential local tempo deviations.

On the other hand, a music score does not describe an actual performance of a given music piece. In particular, it does not describe the actual durations of performed notes in seconds, which we call *physical time*. It also does not describe the actual intensity at which each individual note is played, the variations of tempo, the small deviations of each note relative to the underlying tempo, or possible ornaments that could be added by the musician. In the case of instruments with continuous pitch, it also does not describe the actual performed pitch of a note, for instance when there is vibrato or portamento.

Other types of graphical representations include tablatures (a description of where fingers should be placed on the fingerboard of a string instrument), and extensions of the usual score representations have been designed for contemporary classical music and electronic music (see for instance Kuuskankare [2009]). We will not discuss them here.

### 2.2.2 Symbolic music representations

The above described score format can be seen as an image, that is readable by humans, but not by computers. We describe here computer-readable formats of

symbolic music.

### Digital score formats

First, various encodings exist to digitally represent information written on a score, such as MusicXML [Good, 2001], MEI [Roland, 2002], LilyPond [Nienhuys and Nieuwenhuizen, 2003], or ABC [Walshaw, 2011]. These can precisely represent any score in a digital format. In particular, similarly to a score, they cannot represent a performance of a piece of music.

### MIDI format

Other types of representations exist, that can describe more precisely how a piece was performed and interpreted. One of the most popular machine-readable formats for music representation is the MIDI format [Moog, 1986]. An example is given in Figure 2.2. It is constituted of series of messages, usually separated in tracks. Each message contains an instruction, and a time, which corresponds to the amount of time delay that has to be left between the previous message and the execution of the current. This time delay is given in ticks, which corresponds to fractions of a quarter-note. The time delay is thus given in musical time. The physical duration of a tick is given by the tempo parameter of a MIDI file.

The most common instructions are “note-on” and “note-off”, which indicate that a given pitch, represented as an integer number, has to be activated with a given velocity, or deactivated respectively. Other messages include control changes (for instance, the activation of the sustain pedal), and messages relative to the type of information a MIDI file can hold: tempo changes, time signature changes, key signature changes, or lyrics, among others. However, it is common that information such as key and time signature is left unspecified, or uses default values (C major for key signature and  $\frac{4}{4}$  for time signature) [Raffel and Ellis, 2016].

This format is not equivalent to score formats. Indeed, a pitch is represented by an integer, describing its position on a piano keyboard. Thus, equivalent notes, such as for instance  $C\sharp$  and  $D\flat$ , are represented the same way, while they would be different on a score. Similarly, only the duration of a note can be represented, not the way it is spelled on the score: a quarter note is represented the same way as two tied eighth notes. Ornaments have to be spelled out as individual notes. Generally speaking, all the expression indications of a score have to be translated into specific instructions: in particular, dynamics become velocities, and phrasing has to be expressed by changing note durations.

On the other hand, its temporal precision allows it to represent expressive timing. In particular, although durations are expressed in musical time in a

MIDI file, it often happens that the tempo parameter does not correspond to the tempo of the music it represents. For instance, a MIDI file could be written with a tempo value of 120bpm, but contain music that was performed at a different tempo, with potential tempo variations. The MIDI file is then only used as a description of the music played in physical time only. We call this kind of MIDI files *unquantised*, as opposed to *quantised* MIDI files, where all the durations in ticks exactly correspond to their note values. In particular, some professional renditions of piano pieces can be accurately recorded as unquantised MIDI files, and later reproduced.

### List of notes

A *list of notes* is a representation similar to a MIDI file, but slightly less descriptive. It describes the notes as a list of note tuples  $(s, e, p)$  where  $s$  and  $e$  are the note start and end times, and  $p$  is the note MIDI pitch. An example is given in Figure 2.3. Sometimes, the velocity of the note  $v$  can also be added. In particular, any extra information about time signature, note values, or key signature, is stripped out.

### Piano roll

A *piano roll* is a time-pitch representation, that simply describes whether a note is active at each point in time, discretised as *timesteps*. More formally, it is a binary matrix  $M$  of shape  $N_p \times T$ , where  $N_p$  is the number of pitches considered and  $T$  the total number of timesteps, such that  $M[p, t] = 1$  if and only if pitch  $p$  is active at timestep  $t$ . An example is given in Figure 2.4. Here, the time resolution is determined by the timestep used, while the pitch resolution is usually a semitone, in accordance with Western music notation.

This representation is even less descriptive than the list of notes representation. In such a representation, repeated notes cannot be distinguished from held notes, unless there is at least one timestep of silence between repetitions. For instance, in the list of notes in Figure 2.3, we can see that notes 4 and 5 correspond to a repeated note, while in the piano roll in Figure 2.4, they are represented the same way as a single note with onset time 0.5 and offset time 1s. Besides, the temporal precision of the piano roll depends on its frame rate, while it is arbitrary in a list of notes.

## 2.2.3 Audio representations

In previous representations, notes were represented as discrete symbols. Music can also be represented by its physical manifestation, *i.e.* the resulting sound produced by the instruments. Sound is propagated as acoustic compression

waves. The variation of acoustic pressure in time thus describes the acoustic properties of physical sounds. The value of this acoustic variation through time is called a *waveform*, or an *audio signal*. We write it  $x$ , and it is usually discretised (*i.e.* sampled in time, and quantised in intensity) to be digitised. An example is given in Figure 2.5.

An audio signal can also be converted into a time-frequency representation, using a short-term Fourier transform [Müller, 2015]. Roughly, a spectrogram describes the intensity of each frequency component of the audio signal through time. To do so, the audio signal is first cut into overlapping frames. Then in each frame, the Fourier transform is applied to the audio signal. The time difference between two successive frames is called a *timestep*. We also call the  $t$ -th frame of an audio signal timestep  $t$ . The result is a complex matrix of shape  $F \times T$ , where  $F$  is the number of considered frequency components, and  $T$  is the number of timesteps in the audio signal. More precisely, at any given timestep  $t$ , the frequency component  $f$  (in Hertz) of a short-term Fourier transform of an audio signal  $s$  is written as:

$$STFT(t, f) = \sum_{n=-\infty}^{\infty} x[n]w[n - m]e^{-j\omega n} \quad (2.1)$$

where  $\omega = 2\pi f F_s$ ,  $F_s$  is the sampling rate of the audio signal, and  $w$  is a window (*i.e.* a function with finite time support). The magnitude of this complex matrix is called a *spectrogram*. An example is given in Figure 2.6.

A lot of variants of such representations exist, for instance using logarithmically-spaced frequency components, or logarithmic-scale values for the magnitude of each component. Some more complex time-frequency representations also exist, such as the constant-Q transform [Brown, 1991] or the scattering transform [Mallat, 2012], each with their own properties; we will not detail them here.

## 2.3 Automatic Music Transcription

### 2.3.1 Definition

Automatic Music Transcription (AMT) is the process of converting a music audio signal into a symbolic representation. Most of the time, AMT focuses on converting the audio signal into simple symbolic representations, such as a piano roll or a list of notes. We call Complete Music Transcription (CMT) the task of obtaining a complete staff-notation representation from an audio signal. CMT requires extra steps compared to AMT, such as beat tracking, meter and key estimation, rhythm transcription, pitch spelling, and stream separation. We focus here on AMT.



Figure 2.1: An example music score.

| Message type | Pitch | Velocity | Time (ticks) |
|--------------|-------|----------|--------------|
| NOTE_ON      | 47    | 100      | 0            |
| NOTE_ON      | 62    | 100      | 0            |
| NOTE_ON      | 66    | 100      | 0            |
| NOTE_OFF     | 47    | 0        | 110          |
| NOTE_OFF     | 62    | 0        | 0            |
| NOTE_OFF     | 66    | 0        | 0            |
| NOTE_ON      | 66    | 100      | 110          |
| NOTE_ON      | 69    | 100      | 0            |
| NOTE_OFF     | 66    | 0        | 110          |
| NOTE_OFF     | 69    | 0        | 0            |
| NOTE_ON      | 69    | 100      | 0            |
| NOTE_ON      | 73    | 100      | 0            |
| NOTE_OFF     | 69    | 0        | 110          |
| NOTE_OFF     | 73    | 0        | 0            |
| NOTE_ON      | 66    | 100      | 110          |
| NOTE_ON      | 69    | 100      | 0            |
| NOTE_OFF     | 66    | 0        | 110          |
| NOTE_OFF     | 69    | 0        | 0            |
| NOTE_ON      | 62    | 100      | 110          |
| NOTE_ON      | 66    | 100      | 0            |
| NOTE_ON      | 56    | 100      | 110          |
| NOTE_OFF     | 62    | 0        | 0            |
| NOTE_ON      | 62    | 100      | 0            |
| NOTE_OFF     | 66    | 0        | 0            |
| NOTE_OFF     | 56    | 0        | 110          |
| NOTE_ON      | 56    | 100      | 0            |
| NOTE_OFF     | 62    | 0        | 0            |
| NOTE_ON      | 62    | 100      | 0            |
| NOTE_OFF     | 56    | 0        | 110          |
| NOTE_ON      | 56    | 100      | 0            |
| NOTE_OFF     | 62    | 0        | 0            |
| NOTE_ON      | 62    | 100      | 0            |
| NOTE_OFF     | 56    | 0        | 110          |
| NOTE_ON      | 56    | 100      | 0            |
| NOTE_OFF     | 62    | 0        | 0            |
| NOTE_ON      | 62    | 100      | 0            |
| NOTE_OFF     | 56    | 0        | 110          |
| NOTE_OFF     | 62    | 0        | 0            |

Figure 2.2: An example MIDI file, representing the same note sequence as Figure 2.1, using a tick resolution of 220 ticks per quarter note.

| Note index | MIDI Pitch | Onset time (s) | Offset time (s) |
|------------|------------|----------------|-----------------|
| 0          | 62         | 0.0            | 0.25            |
| 1          | 66         | 0.0            | 0.25            |
| 2          | 47         | 0.0            | 0.25            |
| 3          | 66         | 0.5            | 0.75            |
| 4          | 69         | 0.5            | 0.75            |
| 5          | 69         | 0.75           | 1.0             |
| 6          | 73         | 0.75           | 1.0             |
| 7          | 66         | 1.25           | 1.5             |
| 8          | 69         | 1.25           | 1.5             |
| 9          | 62         | 1.75           | 2.0             |
| 10         | 66         | 1.75           | 2.0             |
| 11         | 62         | 2.0            | 2.25            |
| 12         | 56         | 2.0            | 2.25            |
| 13         | 62         | 2.25           | 2.5             |
| 14         | 56         | 2.25           | 2.5             |
| 15         | 62         | 2.5            | 2.75            |
| 16         | 56         | 2.5            | 2.75            |

Figure 2.3: An example list of notes, representing the same note sequence as Figure 2.1.

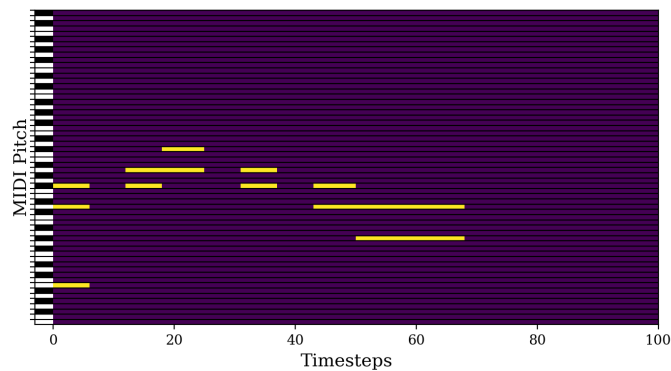


Figure 2.4: An example piano roll, with a timestep of 40ms, representing the same note sequence as Figure 2.1.

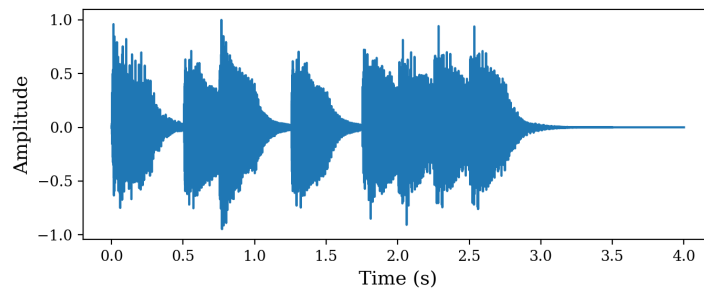


Figure 2.5: An example waveform, representing the same note sequence as Figure 2.1 played on a piano.

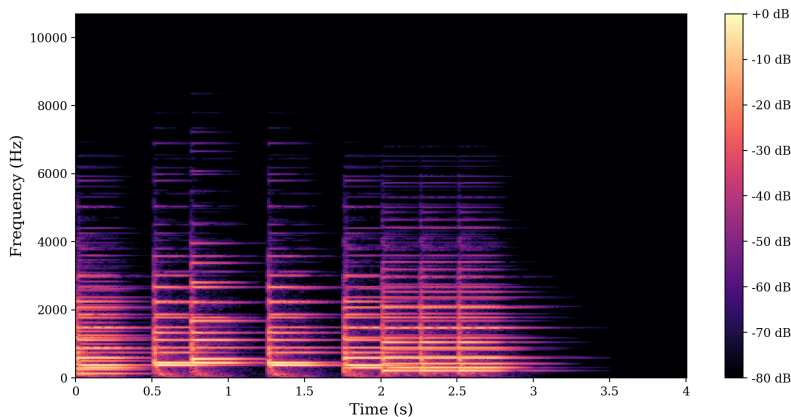


Figure 2.6: An example spectrogram, representing the same note sequence as Figure 2.1 played on a piano.

Usually, a music transcription system takes as input a time-frequency representation (for instance a spectrogram), and outputs a time-pitch *posteriogram*. In the context of AMT, a posteriogram is a real-valued matrix  $P$ , of size  $N_p \times T$ , where  $P[p, t]$  roughly represents the likelihood of pitch  $p$  being active at timestep  $t$ . By analogy with automatic speech recognition (ASR), we call *acoustic model* any system that estimates a posteriogram from audio. A posteriogram can then be discretised into a piano roll as a post processing step. A list of notes can also be obtained, either from the posteriogram directly or from the estimated piano roll; this type of post-processing is also called *note-tracking*. An example of a posteriogram, estimated piano roll and ground truth piano roll is given in Figure 2.7. An overview of this workflow is given in Figure 2.8.

### 2.3.2 Acoustic models

Even though it is not the main topic of this work, we briefly describe the main categories of acoustic models for AMT, along with some notable examples. Acoustic models are also referred to in the literature as multi-pitch estimation models, i.e. models that, as described in Section 2.3.1, estimate at each time frame of an audio signal the set of musical pitches that are being played.

As described by Benetos et al. [2013], there are four main categories of frame-wise multi-pitch estimation models. Some models rely on hand-crafted features, such as pitch-salience functions, or other kinds of functions that rank pitch candidates according to various quality criteria (harmonicity, spectral smoothness, spectral centroid...). For instance, [Su and Yang, 2015] proposes combining estimations given by harmonics in the frequency domain and subharmonics in the lag domain. Some other models use a statistical approach, as in [Davy et al.,

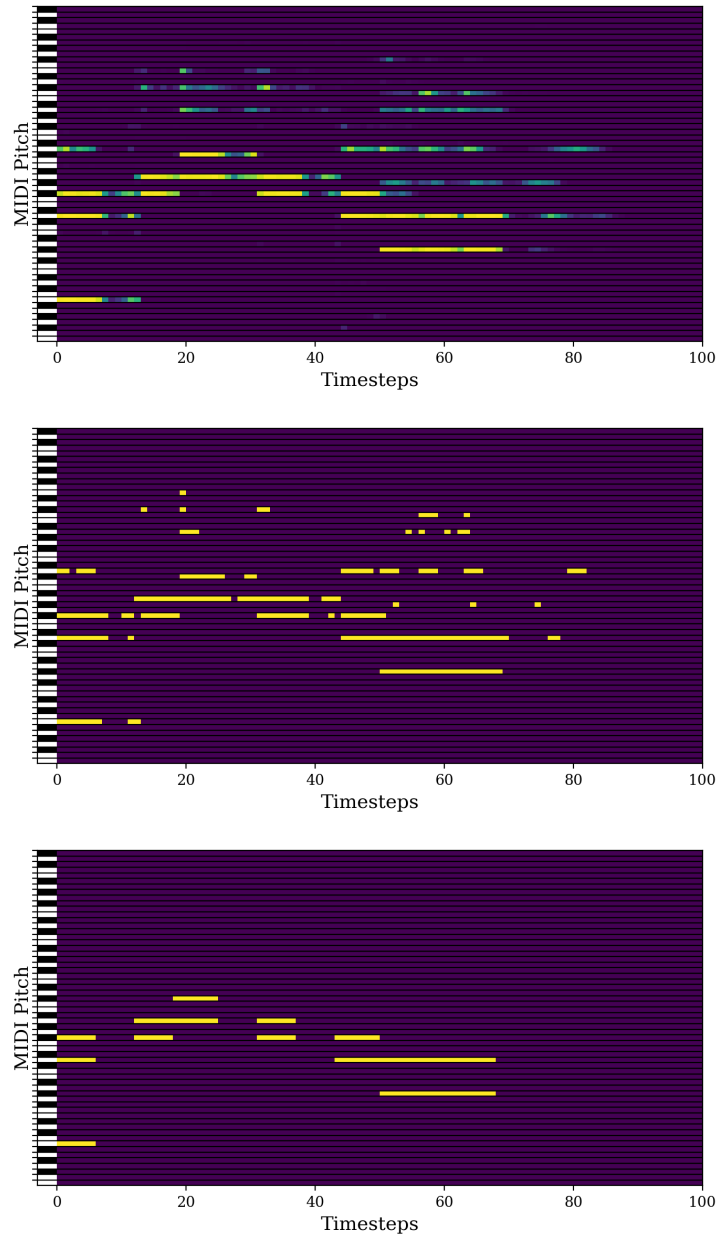


Figure 2.7: An example posterigram, estimated piano roll, and ground truth piano roll (from top to bottom).

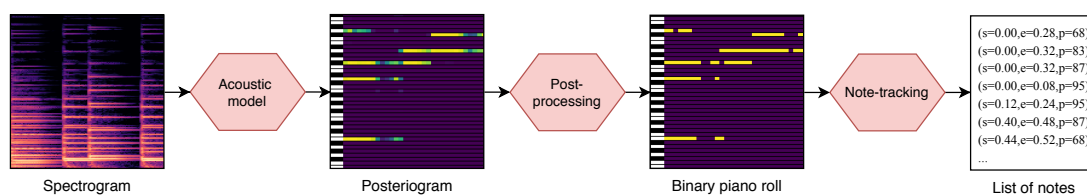


Figure 2.8: Typical AMT workflow.

2006]. The signal or its time-frequency representation is considered as the observation of a probability distribution. The aim is then to try to find the maximum a posteriori (MAP) estimate of its parameters.

More recently, the focus has been put on spectrogram factorisation methods, such as Non-negative Matrix Factorisation (NMF) [Lee and Seung, 1999] and Probabilistic Latent Component Analysis (PLCA) [Hofmann, 1999], its probabilistic counterpart. Those methods, which were first introduced for AMT in [Smaragdis and Brown, 2003], try to decompose the input time-frequency representation matrix into the product of two matrices, one containing note templates, and the other, note activations. The transcription is then obtained from the note activation matrix. Within that framework, Cheng et al. [2016] proposed a refinement focusing on modelling the attack and decay parts of piano sounds more accurately. Another system, described in [Benetos and Weyde, 2015], uses PLCA to decompose an input spectrogram into several probability distributions for pitch activations, instrument source contributions and tuning deviations, using a fixed dictionary of pre-extracted spectral templates.

Finally, neural network approaches have been developing rapidly in recent years, yielding impressive results, despite the lack of datasets of comparable size as image or text processing. In [Kelz et al., 2016], a simple framewise convolutional neural network (CNN) is proposed. It operates on a log-magnitude spectrogram with logarithmically spaced filters, taking as input 5 spectrogram frames, which represents a temporal context of 200ms, and outputs independent presence probabilities for each piano key. In [Bittner et al., 2017], another CNN is proposed for general, multi-instrument framewise multipitch detection, using as input a 3-D tensor representation  $H[h, t, f]$ , stacking harmonically-spaced Constant-Q transform (CQT) along the first axis. This representation allows the network to better detect harmonic content: while in a usual CQT, harmonics of a given sound are spread along the  $f$  axis, in this representation, they are also aligned along the  $h$  axis: if the fundamental frequency of a sound is in bin  $H[0, t, f]$ , its  $k$ -th partial will be found in  $H[k, t, f]$ . In [Hawthorne et al., 2018], two networks are combined: a pitch-wise onset detection module, and a framewise pitch detection module inspired by Kelz et al. [2016], trained

jointly. The combination of both models allows the network to reduce greatly the number of false positives, as a note can only be output if the onset detection and pitch detection agree that both a pitch and an onset are present at a given timestep. Kelz et al. [2019] also propose a multi-task learning approach [Zhang and Yang, 2017] to AMT, meaning that a single network is trained to jointly perform several tasks, usually based on a shared, learnt representation. In this case, the network uses a shared front-end and 3 separate back-ends jointly trained to detect note presence, onset and offsets respectively. The output of these 3 models is then combined using a simple pitch-wise Hidden Markov Model (HMM) enforcing temporal consistency.

### 2.3.3 Music-agnostic post-processing

A posteriogram is real-valued, while a transcription, be it as a piano roll or a list of notes, is discrete. To obtain a transcription from a posteriogram, some post-processing needs to be done. The simplest way to convert a posteriogram into a piano roll is to threshold it. Some simple processing can be applied to make results smoother, for instance median filtering, on the posteriogram or on the piano roll (e.g. in [Su and Yang, 2015]). Other smoothing methods include gap filling, as in [Bello et al., 2006] for instance, and minimum-duration note pruning, as in [Dessein et al., 2010]. A piano roll can then be turned into a list of notes, typically by considering any time transitions from 0 to 1 in the piano roll as a note onset, and vice versa for offsets. It has to be noted that it is theoretically possible to obtain a list of notes as output without first computing a piano roll, it is just a common intermediate step.

Another common, more evolved method was proposed by Poliner and Ellis [2006], and was later used in a variety of systems [Cañadas Quesada et al., 2010, Cazau et al., 2017]. It models each pitch of a piano roll as a two-state, on-off HMM, and treats the posteriogram as observations (a brief general explanation on HMMs is given in Section 2.8.1). The most likely sequence of hidden states can then be decoded with the Viterbi algorithm [Viterbi, 1967]. The initial and transition probabilities are typically learnt from a dataset, with one set of parameters per pitch. Although the parameters are indeed learnt from musical data, since the model does not describe the relation between pitches, we still classify it as music-agnostic: it mostly performs temporal smoothing for each pitch independently. This process is schematised in Figure 2.9. In [Kelz et al., 2019], the post-processing method used is similar to [Poliner and Ellis, 2006], but slightly more refined. Instead of using 2 states (on, off), it uses 4 states (attack, decay, sustain, release), making use of the onset, sustain and offset probability distributions estimated by the acoustic model. In particular, this

method allows the system to explicitly make a distinction between repeated and held notes.

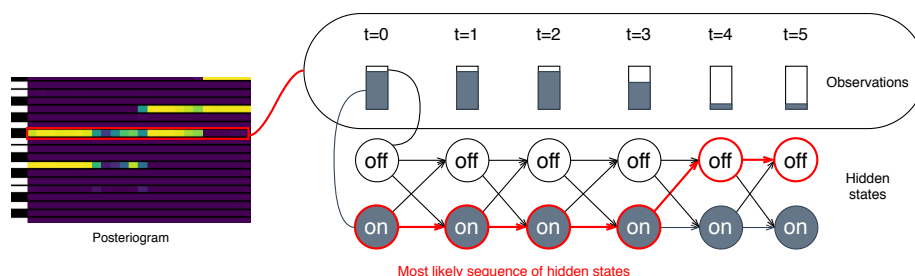


Figure 2.9: A schema of the HMM smoothing method described in [Poliner and Ellis, 2006]. The sequence of activations for each pitch of a posterioqram is seen as observations of a 2-state HMM (on/off). Transition and initial probabilities are obtained from a music corpus. After Viterbi decoding, the piano roll for that pitch has values: 1, 1, 1, 1, 0, 0.

### 2.3.4 Introducing Music Language Models

In all of the above approaches to convert a posterioqram into a symbolic representation, is it considered that music notes are independent from one another (aside from temporal regularities for a single pitch). Yet, it goes without saying that, much like natural language, music is not random: music notes are highly correlated with each other. Contrary to natural language, which is usually monophonic (*i.e.* not more than one word is pronounced at each instant), polyphonic music is made of multiple voices. These voices are internally coherent (each note of a voice is related to the previous notes of this voice, which we call *horizontal dependencies*), and they also interact with each other, forming harmonies (which we call *vertical dependencies*). This structure can be seen at various hierarchical levels, from the scale of a few notes or bars to the scale of sections in a music piece.

In ASR, the equivalent of AMT for natural language, the fact that character or word sequences are highly structured has been extensively used, and greatly contributes to the success of today’s ASR methods [Huang and Deng, 2010], as well as other natural language processing (NLP) tasks such as machine translation or automatic spelling correction. Such models of character or word sequences are called *language models*. By analogy, we call *music language models* (MLMs) symbolic models of music that allow describing this underlying structure. In what follows, we investigate MLMs, focusing on their use for polyphonic music sequence prediction in Chapter 3.

We also look into ways to improve AMT by taking advantage of the ability

of MLMs to model the structure of music sequences. In what follows, we make the distinction between *sequence transduction* and *MLM decoding*.

### Transduction models

Sequence transduction generally refers to the conversion of one sequence into another. In our context, we use it to refer specifically to methods to learn a mapping between a posterioqram (i.e. a sequence of non-binary time-frame vectors), and a piano roll (i.e. a sequence of binary vectors). In that sense, the methods described in Section 2.3.3 can be considered as simple transduction methods. A transduction model is trained on aligned pairs of posterioqrams and piano rolls, and learns a mapping between these two sequences of vectors, in a supervised way. In particular, such a model does not have any explicit knowledge on the distribution of output sequences; it might learn this distribution implicitly, but it cannot estimate the marginal likelihood of a given output sequence. A schema of this workflow is described in Figure 2.10. We investigate this approach in Chapter 4.

There are two main potential aspects a transduction model can exploit to successfully remove mistakes in the posterioqram. One way is to learn the “vertical” dependencies, for instance which notes or note combinations in an input tend to output specific mistakes, and try to correct them. Among these vertical dependencies, some of them can be based on musical aspects (some chords are more likely than others), or on acoustic aspects (some acoustic models tend to output specific false positives, typically octave errors). Another way is to learn the temporal, or “horizontal” dependencies, in other words, given an input sequence, learning which notes are likely to come next. Ideally, a model should take advantage of both of these aspects to improve results.

In most cases, the output of a neural network has to be non-binary, so it can be trained with backpropagation. The output of the network thus has to be further post-processed to obtain a piano roll. In that sense, transduction can be related to image denoising [Fan et al., 2019], as it mostly learns to enhance false negatives and attenuate false positives to produce a cleaner version of the posterioqram. In Section 4.3, we investigate binary neurons (described in Section 2.8.5), which allow directly outputting a binary piano roll without requiring an extra post-processing step.

### MLM decoding

MLM decoding, on the other hand, uses an explicit model of output sequences to assess the likelihood of candidates put forward by the acoustic model, and favour the most likely outputs. Such a model is trained on symbolic data only,

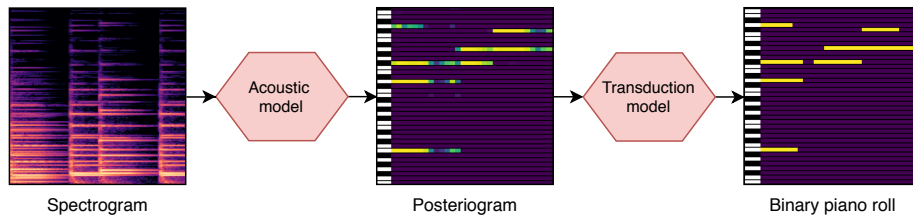


Figure 2.10: Overview of the whole transcription process, using a transduction post-processing step.

which is available in much larger quantities than paired data. In particular, while a transduction model is trained on one specific acoustic model and cannot be assumed to generalise to other acoustic models, in this workflow the MLM is largely independent from the acoustic model. This approach is investigated in Chapter 5.

In the case of MLM decoding, the usual workflow is as follows. At each time frame, some predictions are made by the acoustic model based in the input audio. Then, from these predictions, some possible outputs are selected for the next time-frame. The likelihood of each of these possible continuations is then assessed by the MLM, and the most likely continuations are chosen. Greedily choosing the most likely output at each timestep would result in sub-optimal results. Instead, an approximate global best solution is obtained using a beam search algorithm [Lowerre, 1976] (see Section 5.2.4 for a more detailed explanation of that process), which is an approximate search strategy for the Viterbi algorithm [Viterbi, 1967]. An example of such a workflow is given in Figure 2.11.

## 2.4 Symbolic music modelling

Modelling symbolic music has been a long-running interest among scientists from a variety of domains with various research goals. In this section, we give an overview of some notable symbolic music models.

### 2.4.1 Monophonic vs Polyphonic music modelling

Models of monophonic music can be relatively simple. Markov models [Bishop, 2006] can be used with reasonable success, using for instance one state per possible note. Markov models make the assumption that the likelihood of a symbol only depends on the  $n$  previous symbols (where  $n$  is called the order). A sequence of data can thus be represented as a sequence of  $n$ -grams, *i.e.* a sequence of tuples of  $n$  symbols [Jurafsky and Martin, 2014]. Some more so-

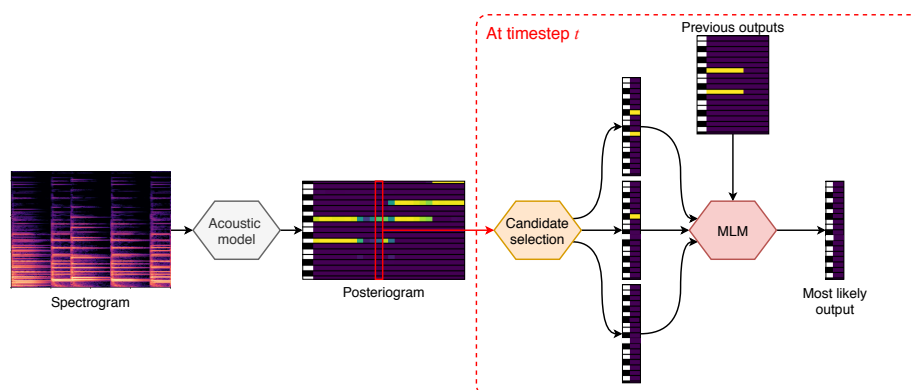


Figure 2.11: An example workflow using framewise MLM decoding. At each timestep, some candidates are selected by the acoustic model, then the MLM assesses their likelihood given the previous outputs, and selects the most likely. The overall best sequence is obtained using beam search [Lowerre, 1976] (not represented, see Section 5.2.4).

phisticated models of monophonic music have been designed. Although it is outside the scope of this thesis to make a comprehensive review of such models, they provide a useful source of inspiration.

Lerdahl and Jackendoff [1983] proposed a Generative Theory of Tonal Music (GTTM), a theory of psychological processing of musical structure, both monophonic and polyphonic, inspired by generative linguistic grammars. Although it was not deterministically specified, it was later partially implemented into a computational model for monophonic music [Hirata et al., 2006, Hamanaka et al., 2017b] and inspired various systems [Nakamura et al., 2016]. Some recent work attempted to adapt this model to polyphonic music [Hamanaka et al., 2017a], by reducing polyphonic music into homorhythmic sequences (*i.e.* polyphonic sequences where all voices use the same rhythm), running the GTTM analysis on the homorhythmic sequences, and expending back the analysis to the original polyphonic music, but this process is not yet automatic.

Conklin and Witten [1995] proposed a system that allows combining multiple features of symbolic sequences (absolute pitch, pitch contour...) called viewpoints, in order to model melodic expectation. IDyOM [Pearce, 2018], a model that stems from the previous, is based on Markov chains using a varying order  $n$ . It uses a multiple-viewpoint framework as in Conklin and Witten [1995], meaning that sequences of notes are described by various features, and analyses are performed on features or groups of features. It allows estimating a probability distribution over possible continuation notes of a given sequence of notes, along with various information theory measures, and was shown to correlate with human expectation of music [Pearce and Wiggins, 2006]. Recent

work applied it to homophonic representations of polyphonic music [Sears et al., 2018].

More recent approaches have used connectionist models, such as [Cherla et al., 2015]. In this paper, an extension of the Recurrent Temporal Restricted Boltzmann Machine [Sutskever et al., 2009] (a variant of the Restricted Boltzmann Machine [Smolensky, 1986] in which hidden units are based on a recurrent neural network) was proposed, and compared favourably with some simpler n-gram based approaches on a monophonic music sequence prediction task. In [Langhabel et al., 2017], a method for feature discovery, essentially combining viewpoints, was proposed for melody prediction.

However, these approaches usually cannot be trivially adapted for polyphonic music. Indeed, for monophonic music, the number of notes is usually of the order of 10 to 100 (a piano has 88 keys). For polyphonic music, all possible chords have to be taken into account ( $2^{88}$  combinations), which is too high a number to be computationally tractable. Some musical knowledge can help reduce this number (*e.g.* usually no more than 10 notes are played at the same time on a piano, and not all combinations of notes are typically found in Western music), but usually the state space remains too big to be explored.

## 2.4.2 Polyphonic music sequence prediction

Among symbolic music models, those that are most relevant for our purpose are models of polyphonic music that allow making predictions as to which notes are likely to come after a given note sequence, a task that we call polyphonic music sequence prediction. This task is not very widely discussed in the literature, and very few systems have been proposed for that specific purpose. We review here some notable examples.

It has to be noted that polyphonic music sequence prediction has been used as an evaluation task in a variety of studies focusing on neural network architectures for sequence modelling [Chung et al., 2014, Jozefowicz et al., 2015, Greff et al., 2017], although their main goal is not music modelling. These studies use polyphonic music prediction as a benchmark task to compare recurrent neural network variants and hyperparameter configurations and find architectures that perform well across the board, not as a goal in itself. In particular, they do not try to improve on this task specifically, nor do they look into how the network behaves depending on the input data.

Some other networks were designed more specifically for music. In [Boulanger-Lewandowski et al., 2012], a neural network architecture combining a Recurrent Neural Network (RNN, see Section 2.8) with a Restricted Boltzmann Machine (RBM) was proposed. In this architecture, the RBM allows modelling the joint

distribution of outputs at a given timestep. The hidden representation of this RBM is obtained from an RNN, which allows linking predictions through time by making the output distribution dependent on the previous pitches. In [Subakan and Smaragdis, 2017], a variant of the Long Short-term Memory unit (LSTM, see Section 2.8.3 for a detailed description) using diagonal recurrent matrices instead of full matrices was proposed and applied to polyphonic music sequence modelling. In other words, in Equation 2.5, the  $N \times N$  matrices  $W_f$ ,  $W_i$ ,  $W_o$ ,  $W$  are replaced by vectors of dimension  $N$ , that are multiplied elementwise with  $h_{t-1}$ . Walder [2016] also proposes using an LSTM network for symbolic music modelling, with an original representation using a variable timestep, where the time difference between two timesteps is an extra variable predicted by the model at each timestep. Lattner et al. [2018] propose a neural architecture that can be trained to explicitly enforce transposition invariance properties in its internal representation. Although it was proposed for monophonic music, it can trivially be adapted to polyphonic music.

All the above approaches are based on corpus analysis: a model is trained on a large dataset and then used to make predictions for new music. Other systems, such as [Walder and Kim, 2018], use music self-similarity to make predictions, based on the idea that in real life, music is very likely to repeat itself with some slight variations. We focus here on the former approach.

### 2.4.3 Other symbolic music models

Many other symbolic music models were developed, but for other purposes, such as human music cognition modelling or music generation. Most of the time, they cannot be applied to our problem as is, but still, some concepts or ideas could be exploited.

Music generation is in itself a widely discussed problem, but a very relevant one for MLMs. Indeed, many of these models implicitly model a probability distribution on possible outputs and use it to draw samples. Comprehensively reviewing this topic is outside the scope of this thesis (see Briot et al. [2019] for a survey). Some systems that manage to combine explicit knowledge of musical priors with data-driven approaches are still worth mentioning. In [Jaques et al., 2017], reinforcement learning techniques [Sutton and Barto, 2018] are used to enforce basic music composition rules in an RNN-based system. Another method, very similar to [Jaques et al., 2017], although much simpler, was proposed for music generation by Sun et al. [2018]. In this paper, they enforce explicit rules using data augmentation, *i.e.* by artificially creating more data to train a neural network. First, an RNN is trained on a given dataset. Then, using this RNN, more data is generated, making sure that the new data abides by

the given rules. A second RNN is then trained on the new dataset, containing both the original data and the generated data. As a result, the rules are more respected in the music generated by the second RNN than by the first. This system is similar to Jaques et al. [2017] in that in both cases, a regular RNN is first trained, and then modified to make sure the generated data follows some given explicit rules. The main difference is that in [Jaques et al., 2017], the modification is made on-line using reinforcement learning, while in [Sun et al., 2018], it is made all at once, during the second training phase.

Some studies have tackled modelling chord sequences and harmony specifically. The System & Contrast Model [Bimbot et al., 2016] provides an interesting multi-scale approach that was applied to chord sequence modelling in [Louboutin and Bimbot, 2016]. Rohrmeier and Graepel [2012] reviewed various models of harmony, namely an n-gram, an HMM and a Dynamic Bayesian Network (DBN), based on a multiple viewpoints framework, and compared their predictive power on a jazz chord progression corpus. They found that different models performed best depending on the viewpoints used as inputs. A more recent comparison of chord sequence models is given in [Korzeniowski et al., 2018], where an n-gram is compared to various RNNs. In this study, the LSTM and Gated Recurrent Unit (GRU) greatly outperform simple n-grams. These models were then used as language models for chord transcription in [Korzeniowski and Widmer, 2018], with an original method combining two separate chord symbol and chord duration models. This allows them to overcome the problem of symbol repetition when the frame rate is high (see Section 2.5.3), but also makes the duration of a symbol independent of its value. This assumption gives good results for chord transcription, as chords tend to change with regular periodicity. However, it would probably not work for AMT, as note durations usually depend on their pitch: out-of-key notes can be present in a piece, but often only as short passing notes.

Other studies tackle the way our brain processes musical content. In [Bigand et al., 2014], a low-level model, based on auditory short-term memory, is compared to higher-level, syntactic-like models. In most cases, the low-level model could account for the results of the experiments in a more parsimonious way than high-level models, partly because the Western musical system is closely related to acoustic properties such as harmonicity. Collins et al. [2014] also compiled various empirical studies on melodic and harmonic expectation, and used the results to develop a model combining both sensory and high-level tonal factors. In particular, they show that combining both types of features gives better results than using only one kind.

## 2.5 Music Language Models for Automatic Music Transcription

Although models designed specifically for music prediction are quite rare, various AMT systems include some kind of model of music sequences in their design. We focus here on music transcription models for polyphonic music, although some monophonic models will be mentioned as sources of inspiration.

### 2.5.1 Probabilistic models

Aside from early attempts to include some musicological priors to transcription systems [Ryynanen and Klapuri, 2005], Temperley [2009] was one of the first to propose a joint probabilistic model for harmony, rhythm and stream separation, and suggested its use for AMT. This model is very interesting conceptually as it is quite comprehensive, explicitly modelling harmony, rhythm and stream separation; however, it seems to be computationally intractable in real-life applications and has not been successfully applied to music transcription.

Since then, many other audio analysis systems have used probabilistic approaches to model high-level musical knowledge in order to improve their performance. Although we focus on neural methods in this thesis, we still give a brief overview of some notable examples. Raczynski et al. [2013] have designed a hierarchical model of harmony using DBNs [Ghahramani, 1997] to post-process the output of an NMF estimator. Benetos [2017] also proposed a model to post-process PLCA posteriors with linear dynamical systems [Kalman, 1962], which can be seen as an extension of HMMs where the hidden space is continuous rather than discrete. In [Simsekli et al., 2013], a multi-level NMF system was proposed, in order to reflect the hierarchical structure of harmony. In particular, this model allows learning from different types of data (audio, symbolic) to train the various parameters. Ojima et al. [2016] also proposed an NMF-based system integrated in a Bayesian framework, that uses as pitch activation prior an HMM modelling chord progressions and pitch distributions. An end-to-end, audio-to-score transcription system was proposed by Kameoka et al. [2012]. It estimates jointly, in a unified Bayesian framework, the notes and the rhythm notation (instead of onsets and offsets in seconds) using probabilistic context-free grammars (PCFGs).

All these approaches are interesting, in particular because they often require little or no data to be trained. However, they often make simplifying assumptions, in particular, in most of these systems, the likelihood of a symbol (for instance a chord or a note) is often assumed to depend only on a finite context (*i.e.* the previous  $n$  symbols). As we argue in Section 2.5.3, temporal dependen-

cies are important in music, and can span much longer time-scales: for instance, it often happens that themes or motives are repeated throughout a piece, several minutes and hundreds of notes apart. As explained in Section 2.8.1, recurrent neural networks have the theoretical ability to overcome that problem, although they still have limited memory in practice.

### 2.5.2 Neural MLMs

More recently, neural networks have been applied to polyphonic music sequence modelling, with the aim of improving AMT performance.

The RNN-RBM model that we mentioned in Section 2.4.2 went on to become fairly popular and was used in several systems. Although it was first proposed as a symbolic model, it was then used for sequence transduction in [Boulanger-Lewandowski et al., 2013], taking as input a Deep Belief Network-based representation of the audio signal. In [Sigtia et al., 2014], the RNN-RBM was used to refine the output of a PLCA multi-pitch estimator, using an iterative workflow: a first estimate is made by the PLCA, then it is processed by the RNN-RBM, potentially removing mistakes, and this new estimate is then used as new initialisation for the PLCA model. In [Sigtia et al., 2016], it was used to decode the outputs of a variety of neural-network-based acoustic models. At each timestep, the likelihood of possible continuations is assessed by the MLM, and the most likely sequence overall is obtained using an approximation of the Viterbi algorithm [Viterbi, 1967] called beam search [Lowerre, 1976] (see Section 5.2.4 for more details).

Another AMT system was proposed in [Wang et al., 2018] using the RNN-RBM, with a difference from the previous systems. Similarly to Walder [2016], this one operates on frames that can have arbitrary durations, corresponding to an inter-onset interval (time interval between two note onsets). Despite being named “note-based”, it does not process lists of note events; it still operates on sequences of binary vectors, like the usual frame-based models, the difference being that the duration of the frames is not constant.

An approach using a sequence-to-sequence framework was proposed in [Ullrich and van der Wel, 2017] for CMT. Here, a CNN encoder is used to extract a compact representation from the audio. This representation is then used to decode the sequences of note symbols. In this approach, the encoder can be seen as an acoustic model, and the decoder as an MLM. While in a usual setup, the MLM and acoustic models are trained separately, this approach has the advantage of being trained end-to-end, using pairs of audio and scores, allowing for co-adaptation of the two models. It was however only proposed as a proof-of-concept model for monophonic music transcription, and has not been

successfully applied to polyphonic music transcription.

### 2.5.3 Limitations

Despite all the work done towards using high-level musical knowledge in AMT systems, many systems reach good performance without using MLMs. In [Kelz et al., 2016], it was shown that [Sigtia et al., 2016] can be outperformed on AMT with neural acoustic models without resorting to the RNN-RBM, simply by carefully tuning hyperparameters and using appropriate input representations (although the authors did not investigate whether using an MLM could further improve results). In [Wang et al., 2018], the MLM is actually only used on very few occasions – when an onset is detected, but the corresponding pitch detection fails – and when it is, it only works with limited success: it tends to fail to predict chords. Using the MLM over the whole note sequence resulted in decreased performance over simple thresholding, possibly due to the discrepancy between training using perfect inputs and decoding noisy sequences (this issue was also noted in [Sigtia et al., 2016]). The current state of the art in AMT is [Hawthorne et al., 2018], a neural model estimating separately the onsets and pitches of note, but using no musicological knowledge – although it cannot be ruled out that some note dependencies may be modelled by the system as a side effect, for instance by the pitch-wise onset detection module. By contrast, in speech recognition, most systems include a language model, and even simple ones such as trigrams can halve the word error rate of a system [Chorowski and Jaitly, 2017]. We outline some limitations of existing models that might explain the lack of effectiveness of MLMs for AMT.

First, most of the above models focus on harmony and melody. The vast majority of them do not take into account anything relative to the temporal organisation of music, be it on the short term (order of a bar), or on the long term (over a whole piece). In very few systems compared to all those cited [Kameoka et al., 2012, Temperley, 2009], is rhythm, meter, and more generally, any temporal feature considered and modeled. In most cases [Benetos, 2017, Sigtia et al., 2016, Şimşekli et al., 2013], it is not considered at all, the models focus on instantaneous, vertical dependencies, and temporal dependencies between notes are only considered on short sliding windows (at most a few notes), and without the notion of absolute metric context. In [Wang et al., 2018], all temporal information is completely discarded, as one frame is used per new onset, regardless of the time between two successive onsets. Besides, in all of the above systems, no aspect of the long-term structure of a piece, such as the repetition of motives, multiple occurrences of a same part (as modelled for instance by [Walder and Kim, 2018]), or the presence of cadences is taken into account.

One key point, that appears to be often overlooked in the literature on frame-based models, is the choice of a relevant timestep. Indeed, in [Boulanger-Lewandowski et al., 2012], when the RNN-RBM was first introduced, it was used with a timestep in fraction of a beat (sixteenth or eighth note). However, in [Boulanger-Lewandowski et al., 2013, Sigtia et al., 2014, 2016], this model was used with a timestep of the order of 10ms, which is both very small compared to a musical duration such as a sixteenth note, and not related to the tempo. As a result, because the same notes are repeated across many timesteps, it is likely that the MLM mostly has a smoothing effect instead of enforcing some kind of long-term musical structure to the output (this is suggested in the conclusion of [Sigtia et al., 2016], but not further investigated). Similar ideas were explored in [Korzeniowski and Widmer, 2017], where a 2-layer LSTM-RNN and HMM were compared on a harmony modelling task. When the frame rate is high (order of 10 fps), the RNN only has a smoothing effect, and is no more efficient than simpler temporal models such as HMMs. However, they suggest that on the chord-level (*i.e.* one symbol per chord, no matter how long), RNNs significantly outperform HMMs. We investigate further this idea throughout this thesis, in Chapter 3 for polyphonic sequence prediction, in Chapter 4 for polyphonic sequence transduction, and in Chapter 5 for MLM decoding.

It has to be noted that in some other MIR tasks, time steps that are not time-constant and depend on the tempo have been proposed. For instance, in downbeat tracking, the task of estimating, given some beat positions, which ones correspond to a downbeat, beat-synchronous features have been used in several systems [Papadopoulos and Peeters, 2010, Durand et al., 2015, Krebs et al., 2016], and studied more specifically by [Fuentes et al., 2018]. For AMT, the only system we found using such musically-relevant timesteps is the one described by [Raczyński et al., 2013], mentioning that they used for audio experiments their symbolic modelling system with an acoustic model using a tempo-synchronous analysis frame size of a sixth of a beat, but they did not elaborate further on that, for instance comparing this time step with a smaller, time-constant time step, or explaining how these time steps were obtained.

The tonal context of the pieces is also rarely taken into account. In [Rohrmeier and Graepel, 2012], tonality is somewhat represented by differentiating for instance C♯ and D♭ (the chords are the same, but the notations occur in different tonalities). In [Boulanger-Lewandowski et al., 2012], it is advised to transpose all the learning sequences in C major/minor as a pre-processing step, which somewhat normalises tonality, although it is not clear how modulations should be managed. But [Raczyński et al., 2013] makes over-simplifying assumptions, such as considering only the intervals between notes or chords. This representation has interesting properties, in particular, being more compact than rep-

representing each pitch individually and being transposition-invariant. However, we argue that it does not fit music, as the likelihood of an interval depends on the starting note and tonality: in C major, an interval of a semitone up is very likely starting from B, but very unlikely starting from C. The rest of the above systems do not take tonality into account and leave it to the system to learn its own tonal representations. Modulations, or finer scale representations such as modes are never modelled.

In most systems, the language model and acoustic model are two rather distinct entities, that interact very little. The worst case is when the MLM is only used as a post-processing step to refine the output of an acoustic model, thus propagating the errors made in the first step [Raczyński et al., 2013, Boulanger-Lewandowski et al., 2012, Benetos, 2017]. In [Sigtia et al., 2014], the two models are also quite independent, but they are integrated in an iterative process, which allows refining the result of the acoustic model thanks to the output of the language model. However, this workflow is not causal, and cannot be used in a real-time system. The system in [Sigtia et al., 2016] attempts to integrate the two models in a causal neural-network-based system. The output of the acoustic model is treated by the language model at each step, instead of being treated all at once. The same kind of workflow is used in [Wang et al., 2018]. Still, in most of the above cases, the acoustic model does not benefit from the language network’s knowledge, and the language model itself does not adapt its behaviour to each acoustic signal. A slightly better option is that described in [Ullrich and van der Wel, 2017], as the two models are trained jointly, but still, the MLM does not inform the acoustic model module.

## 2.6 Evaluation

In this section, we discuss ways to assess the success of our methods.

### 2.6.1 Evaluation of MLMs

Language models can be evaluated in two main ways: by *intrinsic* evaluation or *extrinsic* evaluation [Jurafsky and Martin, 2014].

Intrinsic evaluation refers to the evaluation of a language model on its own, simply on its ability to model symbolic sequences. Extrinsic evaluation refers to the evaluation of the usefulness of a model for a given external task. Intrinsic evaluation is usually easier and more straightforward, as it does not require any external components. However, most of the time, what eventually matters is extrinsic evaluation, as language models are often used to improve other tasks rather than for the sake of it. Ideally, intrinsic evaluation results should be

closely correlated with extrinsic performance.

In NLP, intrinsic evaluation of language models usually means assessing their ability to predict the next character or word of a sequence [Jurafsky and Martin, 2014]. Similarly, here, we investigate polyphonic music sequence prediction to look into the ability of an MLM to model music sequences. In the literature, this is usually assessed by computing the negative log-likelihood, or cross-entropy of a dataset, based on the idea that a good model should give a high likelihood to real data. In our context, given a ground-truth binary piano roll  $M$  of size  $N_p \times T$  and an estimated, non-binary output  $\hat{M}$  of the same size, the cross-entropy is defined as:

$$\mathcal{H}(M, \hat{M}) = - \sum_{t=0}^{T-1} \sum_{p=0}^{N_p-1} M_{t,p} \log(\hat{M}_{t,p}) + (1 - M_{t,p}) \log(1 - \hat{M}_{t,p}) \quad (2.2)$$

This measure is computed for each piece, and then usually averaged across a dataset. This is by far the most common way to compare MLMs intrinsically.

Extrinsic evaluation of MLMs can be done in various ways, depending on the application. MLMs can be applied to automatic music generation, in which case, they can be evaluated by the quality of the generated music. They can also be applied to music cognition modelling, in which case they can be evaluated by how well they correlate with human expectations of music. We use them for AMT, we thus measure their success by the quality of the output transcription. In what follows, we discuss the main evaluation metrics that exist for AMT.

## 2.6.2 Benchmark AMT evaluation metrics

In this section we describe the most commonly-used evaluation metrics for AMT. Some other metrics exist (see [Bay et al., 2009] for a complete description), we only briefly describe here those that are most often used to compare systems.

### Frame-wise metrics

These metrics are computed on pairs of piano rolls. In the MIREX multi-pitch detection task [Bay et al., 2009], a 10ms timestep is used, however this can vary depending on the system considered. When comparing an estimated piano roll  $\hat{M}$  to a target piano roll  $M$ , a true positive is counted whenever  $\hat{M}[p, t] = 1$  and  $M[p, t] = 1$ . False positives and false negatives are counted similarly. We call  $TP$ ,  $FP$  and  $FN$  the total number of true positives, false positives and false negatives, respectively.

The Precision ( $P_f$ ), Recall ( $R_f$ ) and F-Measure ( $F_f$ ) are then computed as

follows:

$$P_f = \frac{TP}{TP + FP} \quad R_f = \frac{TP}{TP + FN} \quad F_f = \frac{2 \cdot P_f \cdot R_f}{P_f + R_f}. \quad (2.3)$$

### Notewise metrics

Notewise metrics are computed on lists of notes. For onset-only notewise metrics, an estimated note  $(\hat{s}, \hat{e}, \hat{p})$  is considered as a true positive if and only if there is a ground-truth note  $(s, e, p)$  such as  $p = \hat{p}$  and  $|s - \hat{s}| < 50ms$ . Besides, ground-truth notes can be matched to at most one estimated note. Precision, Recall and F-Measure (respectively  $P_{n,On}$ ,  $R_{n,On}$ , and  $F_{n,On}$ ) are then computed as in Equation 2.3.

Recently, as  $F_{n,On}$  performance for AMT systems has improved, onset-offset notewise metrics have been increasingly used. They add the extra constraint that, for an estimated note to be considered a true positive,  $\hat{e}$  must be within 20% of the duration of the ground-truth note or within  $\pm 50ms$  of the ground truth offset, whichever is greatest. Again, Precision, Recall and F-Measure (respectively  $P_{n,OnOff}$ ,  $R_{n,OnOff}$ , and  $F_{n,OnOff}$ ) are computed as in Equation 2.3.

In all cases, metrics are computed for each test piece, and then averaged over the whole dataset.

### 2.6.3 Limitations and efforts for better AMT metrics

These metrics are simplistic, in that they mostly count the number of mistakes, but make no distinction between different types of mistakes. Yet, not all mistakes are equally salient to listeners: some mistakes are very noticeable (for instance, out-of-key notes), while some mistakes can go unnoticed even by experienced listeners (for instance, a missing note in a big chord that does not change the resulting harmony). These mistakes affect very differently the perceived quality of a transcription, but they are penalised exactly the same in  $F_{n,On}$ . For framewise metrics, the same criticism can be addressed: repeated notes and held notes would score the same, while being musically very different. Ojima et al. [2016] argued that although their system was not much better in terms of F-measure, it was perceptually better, highlighting the discrepancy between perceptual judgement and benchmark evaluation metrics. However, for lack of better evaluation metric, this affirmation was not quantified.

Recently, various evaluation methods were proposed for CMT [Cogliati and Duan, 2017, McLeod and Steedman, 2018], but they focus mostly on typesetting problems, and do not address the problem of perceptually-relevant pitch assessment. Some efforts were also made for singing voice transcription and melody

estimation [Molina et al., 2014, Bittner and Bosch, 2019], but still consider pitches as being either correct or incorrect. A slightly different approach, based on note patterns rather than individual note detections was proposed in [Frieler et al., 2019]. While this approach allows for more flexibility (a few mistakes might not matter as long as the pattern can be retrieved), it still does not take into account musical aspects such as tonality. It also drops all timing information, which is an important aspect of music (see Section 6.4.1). Another method was proposed for automatic solfège assessment in [Schramm et al., 2016], using a classifier trained on experts ratings to classify each note as correct or incorrect, but again, this decision is mostly binary, and focuses on small deviations in pitch (less than a semitone) rather than the correctness of a pitch in a tonal context.

An older study was conducted on AMT by Daniel et al. [2008]. The study assessed the perceptual discomfort created by some specific types of mistakes (*e.g.* note insertions, deletions, replacement, onset displacement...) by comparing pairs of artificially-modified music excerpts. This data was then used to define new evaluation metrics. However, the types of mistakes considered were relatively limited (for instance, for note insertions, the study only compared octave insertions, fifth insertions and random insertions), and did not take into account musical concepts such as tonality, melody, harmony, or meter. Moreover, the modified MIDI files only contained one type of mistake, and did not consider the potential interactions between several kinds of mistakes.

The evaluation of AMT systems is related to symbolic music similarity, as the end goal is to assess how similar the output is to the target. Symbolic melodic similarity is a widely-discussed problem (see [Velardo et al., 2016] for a survey). Here, we are focusing on polyphonic music similarity, which is much less common. A method for polyphonic music similarity is described by Allali et al. [2009], relying on sequence-to-sequence alignment, using an edit distance function adapted from [Mongeau and Sankoff, 1990]. However, this method was designed for quantised note durations, which makes it potentially suitable for CMT, but not for AMT. We investigate the question of AMT evaluation in Chapter 6.

## 2.7 Available datasets

Since we are aiming to use neural networks, the available data, both in terms of quantity, quality, and diversity, is an important factor of success. We describe here the main datasets available for polyphonic music sequence modelling and AMT.

### 2.7.1 Symbolic music

Various datasets for symbolic music are available. A summary is given in Table 2.1.

The JSBChorales corpus contains all of the 382 four-part harmonized chorales by J. S. Bach. It is not clear who gathered this dataset, the earliest mention we could find is in [Allan and Williams, 2005]. Although the original data is not accessible anymore, it is made available for download, along with pre-processed versions of other datasets by Nicolas Boulanger-Lewandowski<sup>1</sup>. It is interesting because it is very stylistically coherent, but it is also very specific: any model trained on this dataset should probably not be used for other styles.

The Nottingham dataset<sup>2</sup> contains 1200 British and American folk tunes. The original dataset only contains melodies and chord symbols, in ABC format. It was converted into MIDI format by Nicolas Boulanger-Lewandowski<sup>1</sup> by using chord templates, which makes it a fairly simple and consistent dataset.

The Lakh Midi Dataset [Raffel, 2016] contains more than 170,000 MIDI files, of a wide variety of Western music styles, from classical to electronic music, with various instruments (piano solo, multi-instruments). This represents a lot of data, but it is also of lower quality: all styles are mixed, there might be MIDI files corresponding to the same piece, some of them might be corrupt, and the annotations found in the MIDI files can be more or less complete and accurate.

The MuseData dataset<sup>3</sup> contains more than 700 Western classical music pieces, both piano solo and ensemble. For this dataset, full scores are also available. Multitrack MIDI versions of the pieces are also made available by Nicolas Boulanger-Lewandowski<sup>1</sup>.

The Piano-midi.de<sup>4</sup> dataset contains more than 300 Western classical music piano pieces, from a wide range of epochs and composers. While all the above datasets contain quantised data (i.e. data as written in a score, with exact note durations), this dataset was made by manually editing the velocities and the tempo curve of quantised MIDI files in order to give them a natural feeling. It thus contains both quantised durations, and expressive timing. For this reason, we use it for most of our analysis, in particular in Chapters 3 and 5, as it allows us to properly compare the use of time-constant and musically-relevant timesteps in a setup that is as realistic as possible.

In terms of contents, this dataset is skewed towards romantic music (179 pieces, 58% of the dataset), but is fairly balanced in terms of composers: the most represented composer is Frédéric Chopin with 48 pieces (16% of the dataset),

---

<sup>1</sup><http://www-etud.iro.umontreal.ca/~boulanni/icml2012>

<sup>2</sup><https://ifdo.ca/~seymour/nottingham/nottingham.html>

<sup>3</sup><http://musedata.stanford.edu/>

<sup>4</sup><http://piano-midi.de>

| Name              | Size     | Style                           | Multi-instrument | Score | Expressive |
|-------------------|----------|---------------------------------|------------------|-------|------------|
| JSBChorales       | 382      | Baroque chorales by J.S. Bach   | No               | No    | No         |
| Nottingham        | 1200     | British and American folk       | No               | Yes   | No         |
| Lakh Midi Dataset | >170 000 | Western music at large          | Yes              | No    | No         |
| MuseData          | 783      | Western classical music         | Yes              | Yes   | No         |
| Piano-midi.de     | >300     | Western classical music (piano) | No               | No    | Yes        |

Table 2.1: Summary of the main symbolic music datasets. Size is given in number of examples, and does not take into account the duration of each example. “Multi-instruments” describes whether a dataset contains some multi-instrument pieces, “Score” describes whether score-format notation is available, and “Expressive” describes whether the dataset contains pieces with expressive timing.

followed by Ludwig van Beethoven and Franz Schubert with 29 pieces (9% of the dataset) each. It also contains time- and key-signature annotations. The key signature, however, is always given as the major relative (*i.e.* a piece in A minor would be written as C major), and does not take into account short modulations that might occur within a piece without being indicated by a key signature change. Besides, although the tempo curve and velocities vary, it is still different from real performed data. For instance, notes of a chord are always played exactly simultaneously, while in a real performance, there would often be small time deviations, due to the time precision of performers. Moreover, the tempo variations in the MIDI files are usually smaller than in real expressive performances.

### 2.7.2 Aligned MIDI and Audio

Various datasets of music pieces and aligned note information are available. We only report here those that contain polyphonic music, and are big enough to allow supervised training of neural networks. A summary is given in Table 2.2.

For solo piano pieces, the most widely used dataset is the MIDI-Aligned Piano Sounds dataset (MAPS) [Emiya et al., 2010]. It was created by playing MIDI files on a variety of piano synthesisers, as well as on acoustic Disklavier pianos. It contains recordings and ground-truth for isolated piano notes, random chords, usual chords, and music pieces. We focus here on the music pieces only, which represents 270 single recordings, for a total of about 18 hours of music. The synthesised MIDI files were taken from the Piano-midi.de dataset. Thus, as mentioned in Section 2.7.1, they do not correspond to realistic piano

performances, although some effort was made towards making them expressive to some extent. We augment the ground truth for this dataset with beat, meter and key information, which we call A-MAPS and present in Appendix A. We use it in Chapters 4, 5, and 6.

Other datasets are available for piano solo. The Mazurka dataset, compiled by the Centre for the History and Analysis of Recorded Music (CHARM)<sup>5</sup>, contains over 3000 performances of Chopin’s Mazurkas, by various performers, recorded from 1902 to the present day. This dataset is usually used for analysis of expressivity, but could be interesting for AMT, because of its size and wide range of recording conditions. We rule it out because of its lack of diversity in terms of music pieces.

More recently, the MAESTRO dataset [Hawthorne et al., 2019] provides a much bigger dataset of aligned MIDI and audio: more than 1200 single recordings, for a total of 200 hours. In particular, this dataset only contains real, expressive piano performances. However, they do not contain additional annotations, such as rhythm or key.

Some other datasets are available with multiple instruments. The Real World Computing (RWC) dataset [Goto et al., 2002] contains multi-instrument recordings of classical, jazz and popular music. The dataset contains non-aligned MIDI annotations; some automatically-aligned MIDI files are available in the external syncRWC database [Ewert et al., 2009]. Some extra annotations have been proposed throughout the years, such as beat [Goto, 2006] and chords [Cho and Bello, 2011].

The MusicNet dataset [Thickstun et al., 2017] contains 330 pieces of Western classical music, both solo piano and ensembles. It was annotated by automatically aligning scores with audio. The quality of the transcriptions was partly checked by human annotators; its error rate is estimated at around 4%. In particular, annotations also contain note values and metrical positions.

A subset of the Lakh dataset (presented in Section 2.7.1) was automatically matched and aligned to audio files. It contains more than 45,000 pieces, which again is a huge number, but its quality is potentially lower than previous datasets: some imperfect MIDI files could have been matched to the audio, and some transcriptions might be given for irrelevant genres, such as electronic music.

In this thesis, we focus on piano transcription. Moreover, a lot of our investigations revolve around musical timing, so rhythm annotations are important in our case. We thus focus our analyses on the MAPS dataset in the remainder of this thesis.

---

<sup>5</sup><https://charm.rhul.ac.uk/index.html>

| Name              | Size    | Style  | Multi-instrument | Rhythm information |
|-------------------|---------|--|------------------|--------------------|
| MAPS              | 270     | Western classical piano music                      | No               | Yes (see Appx. A)  |
| Mazurka           | 3000    | Chopin's Mazurkas                                  | No               | No                 |
| MAESTRO           | 1200    | Western classical piano music                      | No               | No                 |
| RWC               | 200     | Pop, Jazz, Classical (100, 50 and 50 respectively) | Yes              | Yes                |
| MusicNet          | 330     | Western classical music                            | Yes              | Yes                |
| Lakh Midi Dataset | >45,000 | Western music at large                             | Yes              | No                 |

Table 2.2: Summary of the main AMT datasets. Size is given in number of examples, and does not take into account the duration of each example.

## 2.8 Deep Learning for Sequential and Symbolic Data

In this section, we will define some concepts of deep learning for sequential and symbolic data processing.

### 2.8.1 HMMs and RNNs

Hidden Markov Models (HMMs) are very widely used in many domains [Rabiner and Juang, 1986]. They allow describing the likelihood of a sequence of observations, based on a sequence of discrete hidden states. They make the following simplifying assumptions:

**Output independence:** The likelihood of an observation at a given timestep depends only on the hidden state at this timestep, not on other observations.

**Markov assumption:** The likelihood of a hidden state can be described as a Markov chain. In other words, it only depends on the sequence of the  $n$  previous hidden states ( $n$  is called the order, and is often equal to 1).

The number of hidden states is a parameter of the model. A schema of an HMM of order 1 is given in Figure 2.12. Many variants of HMMs exist, that we will not detail here.

Recurrent neural networks (RNNs) [Rumelhart et al., 1986] are defined as neural networks that are defined with recurrence: they take as input the state of the network at the previous processing step, along with any other kind of input.

They are particularly useful for sequence processing, in particular because they allow modelling temporal dependencies for arbitrary sequence lengths.

The simplest form of an RNN can be described as follows. Given an input sequence  $(x_t)_{0 \leq t < T}$ , the hidden state of an RNN at step  $t$  is expressed as:

$$h_t = \sigma(W_{rec}h_{t-1} + W_{in}x_t + b) \quad (2.4)$$

where  $W_{rec}$  and  $W_{in}$  are trainable weight matrices,  $b$  is a bias vector and  $\sigma$  is the sigmoid function. At each timestep, the hidden state is updated depending on its previous state and the current input. This hidden state can be used as output to the network, or, more commonly, used for further processing. A basic schema is given in Figure 2.13.

This recurrent definition means that the hidden state of an RNN at time  $t$  depends on all the hidden states from time 0 to  $t - 1$ . This represents one of their main differences with HMMs: they can theoretically model dependencies on arbitrary long timescales, while HMMs have a finite memory (a hidden state only depends on a finite number of hidden states). As a result, RNNs can theoretically represent temporal dependencies spanning an arbitrary number of timesteps.

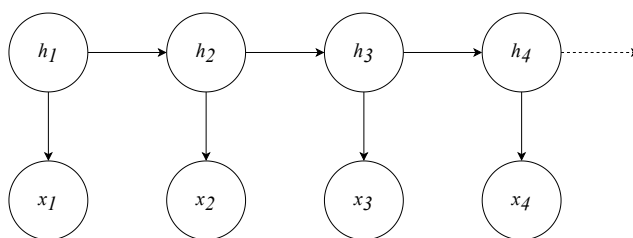


Figure 2.12: Schema of an HMM of order 1.  $x_i$  is the sequence of observations, and  $h_i$  the sequence of hidden states. Arrows represent dependent variables.

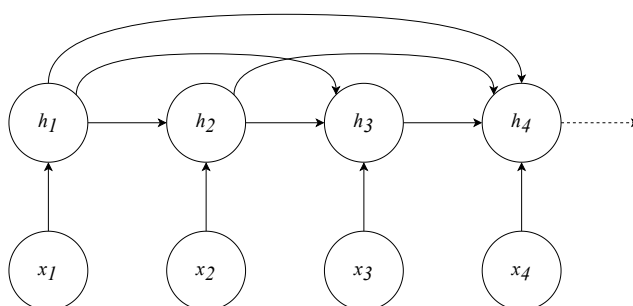


Figure 2.13: Schema of an RNN.  $x_i$  is the sequence of inputs, and  $h_i$  the sequence of hidden states. Arrows represent dependent variables.

## 2.8.2 The vanishing/exploding gradient problem

RNNs in their simplest form are difficult to train. Indeed, given an input sequence of length  $T$ , an RNN can be seen as a deep neural network with  $T$  layers, where all layers have shared weights. Neural networks are usually trained with gradient descent [Rumelhart et al., 1986], meaning that the gradient has to be propagated from the output to the bottom layers. As the gradient is multiplied by a factor at each layer, it grows or shrinks exponentially when  $T$  increases. Thus, it can either vanish (if the factor is smaller than 1), or explode (if it is bigger than 1). This is known as the *vanishing/exploding gradient problem*, and was first detailed by Bengio et al. [1994]. As a result, when the gradient vanishes, the RNN is unable to use information on a scale of more than a few time steps, while when it explodes, training is very unstable. In both cases, training a good model is difficult.

## 2.8.3 Long Short-Term Memory units

As a solution to the vanishing/exploding gradient problem, Long Short-Term Memory (LSTM) units were proposed by Hochreiter and Schmidhuber [1997]. They propose a more sophisticated type of recurrent units. A schema is given in Figure 2.14. It is based on 5 main components:

- A *hidden state* ( $h$ ), similar to regular RNNs
- A *cell state* ( $c$ ), which is essentially a vector into which information can be written, read, and deleted, and carried through timesteps
- Three *gates*, with trainable weights allowing to:
  - Select information from the hidden state and current input that should be added into the cell state (input gate,  $i$ )
  - Select information from the cell state that should be output into the hidden state (output gate,  $o$ )
  - Select information that should be kept or deleted from the cell state at the next time step (forget gate)  $f$ .

Formally, an LSTM unit is defined as follows (biases are omitted for simplicity):

$$\begin{aligned}
 f_t &= \sigma(W_f h_{t-1} + U_f x_t) & c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W h_{t-1} + U x_t) \\
 i_t &= \sigma(W_i h_{t-1} + U_w x_t) & h_t &= o_t \circ \tanh(c_t) \\
 o_t &= \sigma(W_o h_{t-1} + U_o x_t) & &
 \end{aligned}
 \tag{2.5}$$

where  $\circ$  is the elementwise product,  $\sigma$  is the sigmoid function,  $f_t$ ,  $i_t$ ,  $o_t$  are the forget, input and output functions respectively (functions of  $\mathbb{R}^N$ ),  $h_t$  and  $c_t$  are the hidden state and the cell state at time  $t$  respectively (vectors in  $\mathbb{R}^N$ ),  $W_f$ ,  $W_i$ ,  $W_o$ ,  $W$  are trainable weight matrices called recurrent matrices (matrices in  $\mathbb{R}^{N \times N}$ ),  $U_f$ ,  $U_i$ ,  $U_o$ ,  $U$  are trainable weight matrices in  $\mathbb{R}^{N \times I}$ ,  $I$  is the dimension of the input and  $N$  is the dimension of the hidden state. Such matrix multiplications, where the matrix has trainable weights, are also called a dense layer, or a fully-connected layer, as the value of each output depends on the value of all inputs.

By keeping a separate cell state, information can be carried through time, for a theoretically arbitrary number of timesteps, as long as it is not deleted by the forget gate or altered by the input gate.

Since they were first introduced, many variants of the LSTM were proposed. We choose not to detail them here, and refer the interested readers to [Greff et al., 2017] and [Jozefowicz et al., 2015] for a comparison of these variants.

LSTMs are now ubiquitous in sequence processing, and in particular for music sequence processing. They have been used (among other tasks) in AMT [Böck and Schedl, 2012], in melody estimation [Park and Yoo, 2017], in singing voice detection [Lehner et al., 2015] and in music generation [Sturm et al., 2016]. We will investigate their performance for polyphonic music sequence prediction in Chapter 3 and polyphonic music sequence transduction in Chapter 4.

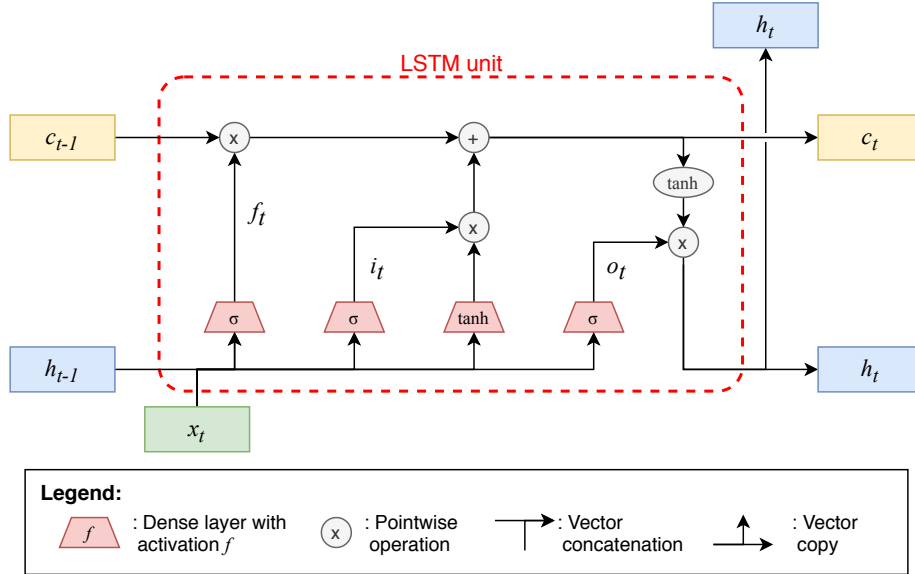


Figure 2.14: Schema of an LSTM unit.

### 2.8.4 Convolutional Neural Networks

Although this thesis focuses mainly on LSTMs, we also consider convolutional neural networks (CNNs) [Goodfellow et al., 2016, Chapter 9] for some tasks.

CNNs are based on convolutional layers. A convolutional layer consists in applying convolutions between the input to the layer, and some filters, whose values are trainable. These convolutions are typically 1-dimensional (1D) or 2-dimensional (2D).

More precisely, in the case of a 1D convolutional layer, in its simplest form, given an input  $(I)_{n \in [1..N]}$ , a filter  $f$  of size  $s$  and an activation function  $\sigma$ , the output  $O_n$  of the layer is expressed as:

$$O_n = \sigma \left( \sum_{i \in [1..s]} f_i * I_{n-s/2+i} \right) \quad (2.6)$$

In other words, the output of the layer is equal to the convolution of the filter  $f$  applied in each point of the input  $I$ . The same idea applies to 2D convolutions, except that the input, output and filters are 2-dimensional matrices instead of 1-dimensional vectors.

The output  $O$  is called a feature map, and typically, each convolutional layer applies several filters  $f$  to the input, each yielding a different feature map. The feature maps are concatenated along an additional dimension (a second dimension for 1D convolutions, a third dimension for 2D convolutions), usually called “channel” dimension. When a convolutional layer is applied to an input with several channels (i.e. a channel dimension  $C$  such that  $C > 1$ ), the filters  $f$  also have a depth of  $C$  along the channel dimension, meaning that equation 2.6 becomes:

$$O_n = \sigma \left( \sum_{c \in [1..C]} \sum_{i \in [1..s]} f_{i,c} * I_{n-s/2+i,c} \right) \quad (2.7)$$

One of the main difference between CNNs and RNNs resides in their receptive field, i.e. the number of input points each output point has access to. CNN outputs, in the absence of fully-connected layers, can only exploit correlations between points located in a finite zone. The extent of this zone is determined by the size of the filters, the number of layers, and various other parameters (such as the amount of overlap between convolutions within each layer, or the presence of downsampling layers such as max pooling). On the other hand, RNN outputs can theoretically exploit information contained in any part of their past, without size restrictions.

Now ubiquitous in deep learning, CNNs were first introduced for image processing, where they have been particularly successful, being directly involved in

many breakthroughs in tasks such as handwritten character recognition [LeCun et al., 1998] or image classification [Krizhevsky et al., 2012]. Their efficiency for such tasks rests on their ability to recognise objects in various parts of a picture: while a feed-forward network would perceive a translated version of an image as a completely different image, a CNN would be able to reuse the same filters, just applied in a different location.

CNNs have also been successfully applied in many MIR tasks, such as audio tagging [Pons et al., 2018], source separation [Jansson et al., 2017], multi-pitch estimation [Bittner et al., 2017], and chord recognition [Korzeniowski and Widmer, 2016]. In the above, 2D CNNs are used, operating on time-frequency representations seen as images. More recently, 1D CNNs have become popular, operating directly on waveforms. The WaveNet architecture [van den Oord et al., 2016], based on multiple layers of dilated convolutions allowing to aggregate a receptive field of several seconds while keeping a low number of parameters, has been particularly influential, and was later used in tasks such as sound synthesis [Engel et al., 2017] or source separation [Stoller et al., 2018].

### 2.8.5 Binary neurons

The output of neural networks is typically real-valued to enable training by gradient descent. Indeed, gradient descent requires that the gradient of the output should be defined in every point. However, symbolic data such as a piano roll is discrete, which means that its gradient is not defined. When, in our case, the objective of a system is to output a piano roll, this has to be done in two steps: first, the network computes a non-binary piano-roll estimation, then some post-processing (typically, thresholding) has to be applied. However, this can be problematic: for piano rolls, it can lead to overly-fragmented notes when the output values are close to the threshold (see Section 4.2). Binary neurons [Bengio et al., 2013] offer an alternative by integrating the binarisation of outputs into the training process while still allowing gradient back-propagation. That way, the model is trained directly with binary outputs, and does not require an extra thresholding step. They were used in the context of music in [Dong and Yang, 2018], in a piano-roll generation system using Generative Adversarial Networks (GANs) [Goodfellow et al., 2014].

Usually, binary neurons work as follows. They receive a real-valued input  $x$ , that is trainable and allows gradient back-propagation. Then, in the forward pass, they apply a discretisation operation, which is non-differentiable. This discretisation operation is usually of two kinds: stochastic or deterministic. With stochastic binary neurons (sBNs), the binary output is obtained by sampling

from a Bernoulli distribution of parameter  $x$ :

$$sBN(x) \sim \mathcal{B}(\sigma(x)) \quad (2.8)$$

where  $\sigma$  is the sigmoid function.

With deterministic binary neurons (dBNs), the binary output is obtained by applying a threshold function to  $x$ :

$$dBN(x) = u(\sigma(x) - 0.5), \quad u(y) = \begin{cases} 1 & \text{if } y \geq 0 \\ 0 & \text{if } y < 0 \end{cases} \quad (2.9)$$

Various methods have been proposed to overcome the fact that the discretisation operation blocks gradient back-propagation, such as REINFORCE [Williams, 1992, Mnih and Gregor, 2014], and the straight-through estimator [Hinton et al., 2012b, Bengio et al., 2013]. We focus here on the straight-through estimator because it was shown to yield good performance for piano-roll generation in [Dong and Yang, 2018]. The straight-through estimator is conceptually very simple: it corresponds to replacing the discretisation operation by the identity function in the backward pass. A modified version to this estimator, the sigmoid-adjusted straight-through estimator, was proposed in [Chung et al., 2017], replacing the discretisation operation by a sigmoid function in the backward pass.

Deterministic binary neurons with the sigmoid-adjusted straight-through estimator are reported as improving over a similar system with non-binary outputs [Dong and Yang, 2018] by being easier to train and yielding better results. We investigate them for polyphonic music sequence transduction in Section 4.3.

## Chapter 3

# Polyphonic Music Sequence Prediction

### 3.1 Introduction

In this chapter, we aim to study the performance of MLMs. We investigate their intrinsic performance, and the relation between intrinsic and extrinsic performance in the context of AMT. In NLP, language models are usually evaluated on a prediction task [Jurafsky and Martin, 2014], in other words, trying to predict the next character or word in a sequence. Similarly, we evaluate here neural models on a polyphonic music prediction task.

Instead of building increasingly sophisticated architectures hoping to obtain better results, we propose to investigate comprehensively and systematically the performance of a simple LSTM network. By studying its behaviour experimentally, we aim to gain a deeper understanding of LSTM models' empirical behaviour when used for polyphonic music sequence prediction, their strengths and shortcomings. In particular, we propose various metrics that can be used as diagnosis tools in order to obtain qualitative insights into what models manage or fail to do in the context of music sequence modelling, along several dimensions. We also propose a new parametric training loss based on the previous metrics, allowing us to adjust the behaviour of the model for a specific task. Finally, we look into the relation between loss parameters, intrinsic MLM performance and extrinsic MLM performance when used for AMT, and in particular, we investigate to what extent intrinsic MLM performance is correlated with extrinsic AMT performance.

The remainder of this chapter is organised as follows. In Section 3.2, we present the experimental setup we will use throughout our experiments. In Sec-

tion 3.3, we present the results of a preliminary experiment comparing various timestep configurations with benchmark metrics. In Section 3.4, we formulate all the evaluation metrics, both benchmark and newly-proposed, that we will use in our further experiments, and we combine them into a new parametric loss to train our models. In Section 3.5, we present the results of our experiments using all the evaluation metrics, we compare our model against benchmark prediction systems found in the literature, and we investigate the influence of loss parameters on prediction performance, as well as on AMT performance as an extrinsic task. Finally, in Section 3.6, we discuss the accomplished work and propose new directions for developing neural network-based prediction models.

The work presented in this section was accepted to the IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP), reference (10) from Section 1.4. It extends work presented at the International Society for Music Information Retrieval Conference (ISMIR 2017), reference (3) from Section 1.4.

## 3.2 Experimental setup

### 3.2.1 Problem statement

We study the performance of an LSTM network on the task of polyphonic music sequence prediction, as is usually done to evaluate language models in NLP [Jurafsky and Martin, 2014]. We use a frame-based model, operating on piano rolls. This choice was made as we believe it allows simpler and more direct integration with frame-based multi-pitch detection systems. Formally, a piano roll is a  $N_p \times T$  matrix  $M$ , where  $T$  is the number of timesteps, and  $N_p$  is the number of considered pitches. Here, we use  $N_p = 88$ , with each pitch corresponding to a key on a piano, between MIDI notes A0 and C8.  $M$  is binary, such that  $M[p, t] = 1$  if and only if pitch  $p$  is active at timestep  $t$ . In particular, held notes and repeated notes are not differentiated. The output of our model is of the same form, except that the values are non-binary between 0 and 1, and can be interpreted as independent probabilities of each pitch being active at each timestep. More specifically, let  $M_t$  be the 88-dimensional vector corresponding to the  $t$ -th timestep of  $M$ . For all  $t \in \llbracket 0, T - 1 \rrbracket$ , we try to predict  $M_{t+1}$  given the ordered sequence of  $\{M_i\}_{i \in \llbracket 0, t \rrbracket}$ . We call the non-binary predicted matrix  $\hat{M}$ , and the binarised predicted matrix  $\tilde{M}$ . We index  $\hat{M}$  and  $\tilde{M}$  by  $t \in \llbracket 1, T \rrbracket$ , such that a perfect prediction system would get  $\tilde{M}_i = M_i$ .

We use three different kinds of timesteps:

- *time-based* timesteps, that have a fixed duration in milliseconds (for instance 10ms).

- *note-based* timesteps, that have a fixed musical duration (for instance, a sixteenth note). In this case, each timestep has a different physical duration in milliseconds. In practice, using note-based timesteps normalises the dataset with respect to tempo.
- *event-based* timesteps, where timesteps correspond to new notes and chords, regardless of their durations.

To obtain a piano roll from a MIDI file, we build a list  $\mathcal{T}$  corresponding to the times in seconds of the timesteps (*e.g.*  $\mathcal{T} = [0, 0.01, 0.02\dots]$  for 10ms timesteps). We then consider that a note is active at a given timestep if it is active in the MIDI file at the start of this timestep. This might lead to unnatural results for instance with 16th note timesteps if, as is the case in our dataset, there are slight imprecisions in the note onset and offset times (*e.g.* if a note starts or ends slightly after a 16th note position). To account for that, we allow some tolerance, both for onsets and offsets, with different thresholds, meaning that if a note starts just after a timestep, it will still be considered active at that timestep, or conversely, inactive if it ends just after. Let  $N = (p_n, s_n, e_n)_{0 \leq n < N_n}$  the sequence of notes in a given piece, where  $N_n$  is the number of notes and each note  $n$  has pitch  $p_n$ , start time  $s_n$  and end time  $e_n$ . Given tolerance thresholds  $T_s$  and  $T_e$  for onsets and offsets respectively, we have:

$$M[p, t] = 1 \Leftrightarrow \exists n | p_n = p \wedge (s_n \leq \mathcal{T}_t + T_s) \wedge (e_n > \mathcal{T}_t + T_e) \quad (3.1)$$

In practice, we define  $T_s$  and  $T_e$  as percentages of the duration of the current timestep. For 10ms timesteps, we choose  $T_s = T_e = 0$ , since the timestep is small enough that displacements by 1 timestep are negligible. For other timesteps, we compare the resulting piano roll, upsampled back to 10ms timesteps to the original 10ms-timestep piano roll, and choose the values that maximise the framewise F-Measure over the whole dataset between these two piano rolls through grid search.

### 3.2.2 Model

Our primary goal is to study the behaviour and potential of a simple RNN architecture. In order to avoid being influenced by other parameters, we deliberately choose to use a simple configuration of the most widely-used RNN architecture, the LSTM. In particular, we choose not to use several layers, nor to use dropout or any other regularisation method during training. We make the code for our model available for future use<sup>1</sup>.

<sup>1</sup><https://github.com/adrienycart/PolyMusicPredLSTM>

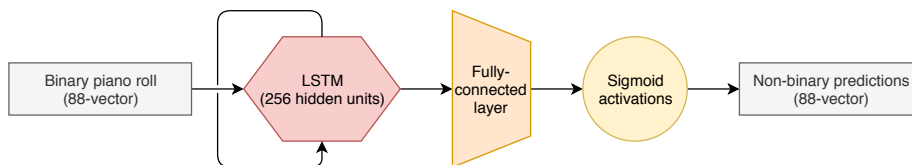


Figure 3.1: Single-layer LSTM network architecture. The non-binary predictions can then be thresholded to obtain binary predictions.

We thus use an LSTM with 88 inputs, one single hidden layer with  $N_h$  hidden nodes, and one fully-connected layer with 88 outputs, one for each piano key, which are sent through a sigmoid function. The network architecture is shown in Fig. 3.1. We use as cost function the cross entropy between the output of the sigmoid and the ground truth ( $\mathcal{H}$ , as defined in Section 3.4.1), as is typically done. To set the number of hidden nodes and the learning rate, we try the following parameters, as a simpler alternative to extensive grid search:  $N_h \in \{128, 256\}$  and  $l \in \{0.001, 0.01\}$ . We find that results are very similar in all configurations (all within 1% of each other), except with *event* timesteps (see Section 3.3), where the configuration  $N = 256, l = 0.01$  was slightly better than the others. For the rest of the experiments, we keep this hyper-parameter configuration.

We train the models for a maximum of 500 epochs and use early stopping, such that if the training loss computed on the validation dataset does not decrease for 15 epochs, we stop training and keep the last best model. In practice, all models stopped training before reaching the 500 epochs limit. The output of the network is then thresholded to obtain a binary piano roll. The threshold is determined by choosing the one that gives the best results on the validation dataset (see section 3.2.3), and we use one single threshold for all pitches.

### 3.2.3 Dataset

#### The Piano-midi.de dataset

We use the Piano-midi.de dataset<sup>2</sup> as real-world MIDI data (see Section 2.7.1 for a longer description). This dataset currently holds 307 pieces of classical piano music from various composers. It was made by manually editing the velocities and the tempo curve of quantised MIDI files in order to give them a natural feeling. We can thus have access to quantised durations to compute note-based timesteps, and expressive timing for time-based timesteps. Besides, it also contains time- and key-signature annotations. The key signature, however, is always given as the major relative (*i.e.* a piece in A minor would be written as

<sup>2</sup><http://piano-midi.de/>

C major), and does not take into account short modulations that might occur within a piece without being indicated by a key signature change. Another motivation for using this dataset is that the MAPS dataset [Emiya et al., 2010], a widely used benchmark dataset for AMT, was created using MIDI files from the Piano-midi.de dataset. By using this dataset, we can make our experiments in a context as close as possible to how our system would be tested for AMT, and thus hope to have results more directly transferable to AMT.

This dataset holds pieces of very different durations, from 20 seconds to 20 minutes. In order to be more computationally efficient, we only keep the first minute from each file and we zero-pad the shorter files. The resulting dataset is 5 hours long. We split it into training, validation and test datasets with the following respective proportions: 70%-10%-20%. The exact split is made available<sup>3</sup>.

### Data augmentation and pre-processing

Many musical concepts (*e.g.* modes, chord types, cadences...) are defined in terms of pitch intervals rather than absolute pitches. In particular, they are not affected by transposition; it is thus useful to enforce such transposition-invariance properties in an MLM. To do so, we consider two options. The first one, that we call *transpose C*, consists of transposing every piece as a whole into C major (we remind the reader that the minor keys are always written as their major relative in the dataset annotations), as per [Boulanger-Lewandowski et al., 2012]. When there are key modulations, we choose as key signature the one that lasts longest, and transpose the piece into C major accordingly. This keeps the size of the dataset as is. Another option, that we call *transpose all*, is to transpose every piece in every key, from 7 semitones below to 5 semitones above. That way, all tonalities are equally represented; this increases the size of the dataset 12-fold.

We compare these two approaches in the preliminary experiment (we do not present detailed results for the sake of conciseness). While both approaches improve the results, *transpose all* yielded greater improvement. It is also simpler, as it does not require knowing the key of the pieces in advance. Besides, when testing a model trained with *transpose C* on data that is not transposed (keeping the original tonality), results are actually worse than with no data augmentation at all. We thus use *transpose all* in all further experiments.

---

<sup>3</sup><http://c4dm.eecs.qmul.ac.uk/ycart/tas1p20.html>

### 3.3 Preliminary experiment

#### 3.3.1 Description

As a first experiment, we study how the choice of a timestep influences the performance of the MLM. More specifically, we compare 5 timesteps:

- *time-short*: 10ms (typical timestep for audio analysis)
- *time-long*: 180ms (average duration of a 16th note)
- *note-short*: a 48th note (greatest common divisor of most usual musical durations)
- *note-long*: a 16th note (typical musical duration)
- *event*: one timestep per new onset

For each of these timesteps, we evaluate two models:

- *LSTM*: the LSTM described in Section 3.2.2.
- *Baseline*: a naive baseline model, that simply repeats the previous output, in other words, a system such that  $\hat{M}_t = M_{t-1}$ . To be able to compute the cross entropy, we replace ones with 0.999 and zeroes with 0.001.

In all cases, a system is trained and tested using the same timestep. The systems are evaluated using the benchmark metrics presented in Section 3.4.1, namely F-Measure ( $\mathcal{F}$ ), precision ( $\mathcal{P}$ ), recall ( $\mathcal{R}$ ), all computed on binary outputs, and cross entropy ( $\mathcal{H}$ ), computed on sigmoid outputs. Comparing *time-long* and *note-long*, we can study the influence of using note-based timesteps. We can also study the influence of the size of the timestep with both time-based and note-based timesteps by comparing *time-short* and *time-long*, and *note-short* and *note-long* respectively. The *event* configuration echoes the chord-level configuration in [Korzeniowski and Widmer, 2017], the difference being that there might be repeated frames when the same chord is played twice.

It has to be noted that the size of the timestep determines the temporal precision of our model. For instance, using a 16th note timestep makes it impossible to accurately represent durations that are not a multiple of a 16th note. Using a timestep of 180ms thus is not advisable in a real case scenario, and we only consider this timestep for comparative purposes.

#### 3.3.2 Quantitative analysis

Results for this preliminary experiment are reported in Table 3.1. It appears as a general trend that the longer the timestep, the bigger the difference in

| Model           | $\mathcal{F}$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{H}$ |
|-----------------|---------------|---------------|---------------|---------------|
| TIME-SHORT      |               |               |               |               |
| <i>Baseline</i> | <b>96.7</b>   | <b>96.7</b>   | <b>96.7</b>   | 1.25          |
| LSTM            | <b>96.7</b>   | <b>96.7</b>   | <b>96.7</b>   | <b>0.892</b>  |
| TIME-LONG       |               |               |               |               |
| <i>Baseline</i> | <b>61.6</b>   | <b>61.6</b>   | 61.6          | 15.2          |
| LSTM            | 61.4          | 60.9          | <b>62.3</b>   | <b>6.09</b>   |
| NOTE-SHORT      |               |               |               |               |
| <i>Baseline</i> | 83.8          | 83.8          | <b>83.8</b>   | 5.78          |
| LSTM            | <b>84.1</b>   | <b>86.3</b>   | 82.0          | <b>2.76</b>   |
| NOTE-LONG       |               |               |               |               |
| <i>Baseline</i> | 60.8          | 60.7          | 60.8          | 15.1          |
| LSTM            | <b>63.3</b>   | <b>64.8</b>   | <b>62.3</b>   | <b>5.4</b>    |
| EVENT           |               |               |               |               |
| <i>Baseline</i> | 41.1          | <b>41.0</b>   | 41.2          | 24.1          |
| LSTM            | <b>46.2</b>   | 40.6          | <b>53.8</b>   | <b>7.77</b>   |

Table 3.1: Prediction performance for various timesteps assessed with the benchmark metrics (defined in Section 3.4.1). Bold values correspond to the best result for each timestep, and we underline it when it is significantly better (i.e.  $p < 0.05$  with a paired t-test).

performance between the naive baseline model and the LSTM. Indeed, when the frame rate is higher, a given note spans more timesteps. As a result, there are many more self-transitions (ie. two identical successive frames) than note changes. The network thus learns that most of the time, a pitch active at timestep  $t$  will still be active at  $t+1$ . In the *time-short* configuration, this effect is particularly visible: there is nearly no performance difference between the LSTM and the baseline model. Repeating the previous pitches constitutes a very good strategy in this case, as shown by the very good prediction performance, but that does not make a good model of music. On the other end of the spectrum, in *event* configuration, the LSTM performs much better than the baseline model, as in this case, there are fewer self-transitions. This observation is in accordance with the findings in [Korzeniowski and Widmer, 2017].

### 3.3.3 Qualitative analysis

When inspecting the non-binary outputs of the networks, some interesting differences between note-based and time-based timesteps can be noticed. We provide some examples of outputs for all the configurations in Figure 3.2.

With note-based timesteps, it appears that every few timesteps (every 4

timesteps *i.e.* a quarter-note in *note-long* configuration, 6 *i.e.* an eighth-note in *note-short* configuration), the network lightly activates some outputs that are different from the previous ones, and deactivates the previous one. The network thus has learned that a transition might occur at these timesteps, which is very sensible, given that note transitions occur more frequently on beats and half-beats. This shows that the network has learned some kind of representation of temporal periodicities in music (which form an important basic component of meter). This hinders prediction performance, as predicting a note change when there is none is a mistake, but constitutes an interesting feature from a music modelling perspective. It has to be noted that this behaviour does not happen in the *time-long* configuration, which shows that it does not simply come from longer timesteps. In *time-short* configuration, the system exhibits this behaviour to a much lower degree, which proves that despite the fact that notes might last an arbitrary number of timesteps, the LSTM is able to pick up some regularities and exploit them to make predictions.

When comparing the effect of having short timesteps, we see that the network tends to be much more confident in its predictions, especially when predicting that a note is going to be repeated. Indeed, we can see that the “background noise” of the pictures, *i.e.* the bins that do not correspond to correct note predictions, have very low values. On the other hand, with longer timesteps, the background has higher values, meaning that the network is less confident. There are also some phantom notes that are not in the ground truth, that correspond to the notes of the scale of the piece. This shows that the network was able to infer, from the first few notes, which pitches are likely to occur, based on the occurrence of similar note patterns in the dataset. This is particularly visible in *time-long*, *event* and *note-long* configurations. Again, this has a negative effect on prediction performance, but having an idea of what notes are likely to occur is a desirable quality for an MLM.

The main problem that emerges from this preliminary experiment is that the usual metrics, although still informative, are not sufficient to capture the fitness of a model for musical purposes. Indeed, a very good prediction performance is obtained even by the baseline model in the *time-short* configuration, but all the system is doing is repeating the previous time-step. On the other hand, note-based timesteps seem to have interesting properties, like inferring when notes might change, but these properties are not shown in the metrics used, they are even penalised. We thus design metrics that will hopefully help highlight interesting musical properties.

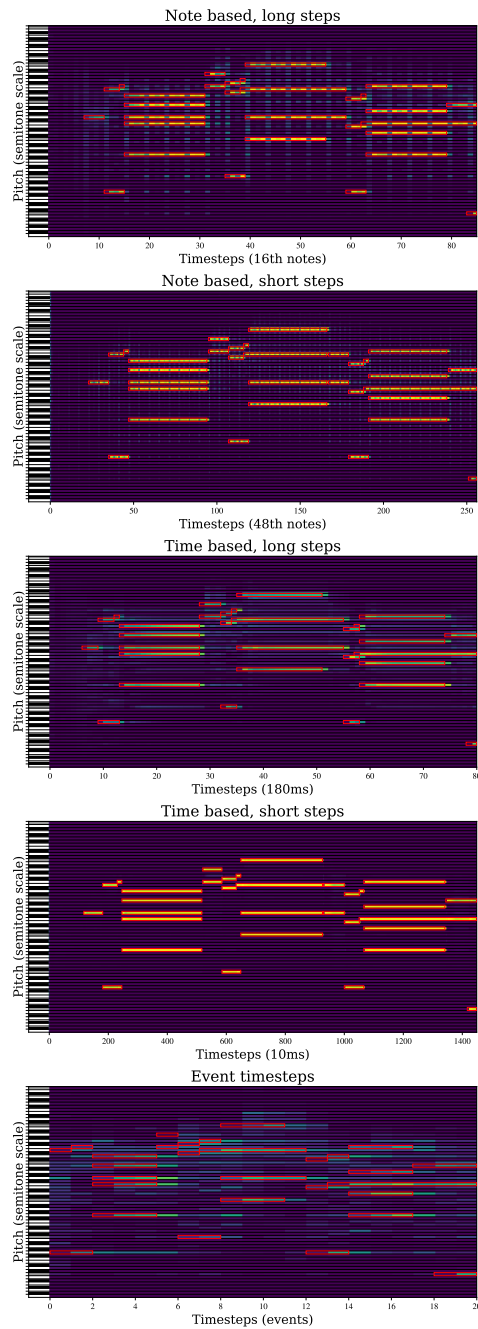


Figure 3.2: Preliminary experiment: Comparison of sigmoid outputs for various timesteps for the MIDI file `chpn-p7.mid` transposed into C. Ground truth notes are overlaid as red rectangles. The notes of the scale correspond to the white keys on the left of each image. The tonic is in grey. The same color map is used across figures.

### 3.4 Evaluation metrics

#### 3.4.1 Benchmark evaluation metrics

In this section, we present the evaluation metrics we use to compare model performance. First, we compute several metrics following the MIREX Multiple-F0 Estimation task [Bay et al., 2009], namely the frame-wise precision, recall and F-Measure. These measures are defined similarly to Section 2.6.2, that is to say:

$$\mathcal{P} = \frac{\sum_{t=0}^T TP(t)}{\sum_{t=0}^T TP(t) + FP(t)} \quad (3.2)$$

$$\mathcal{R} = \frac{\sum_{t=0}^T TP(t)}{\sum_{t=0}^T TP(t) + FN(t)} \quad (3.3)$$

$$\mathcal{F} = \frac{2 \cdot \mathcal{P} \cdot \mathcal{R}}{\mathcal{P} + \mathcal{R}} \quad (3.4)$$

where  $TP(t)$ ,  $FP(t)$  and  $FN(t)$  are the number of true positives, false positives and false negatives, respectively, when comparing  $\tilde{M}_t$  and  $M_t$ . We count one true positive for each  $(p, t)$  such that  $\tilde{M}[p, t] = M[p, t] = 1$ . While the definition of  $\mathcal{F}$ ,  $\mathcal{P}$  and  $\mathcal{R}$  is similar to  $F_f$ ,  $P_f$  and  $R_f$  as defined in Section 2.6.2, we use different symbols to make a distinction between prediction metrics and AMT metrics.

We also use the cross entropy between the sigmoid output of our network and the binary targets. The cross entropy  $\mathcal{H}(M_t, \hat{M}_t)$  between the vectors  $M_t$  and  $\hat{M}_t$  is defined as:

$$\mathcal{H}(M_t, \hat{M}_t) = - \sum_{p \in [0, 87]} M[p, t] \log(\hat{M}[p, t]) + (1 - M[p, t]) \log(1 - \hat{M}[p, t]) \quad (3.5)$$

The cross entropy measure of two sequences of vectors is defined as the average of the cross entropy across timesteps.

Those metrics are computed for each piece, and then averaged over a dataset. We abbreviate precision, recall, F-Measure, and cross entropy as  $\mathcal{P}$ ,  $\mathcal{R}$ ,  $\mathcal{F}$ , and  $\mathcal{H}$  respectively.

#### 3.4.2 Proposed evaluation metrics

In order to get deeper insight into the performance of MLMs, we propose additional metrics, that we describe in what follows. For all these metrics, lower is better.

### Transition Cross entropy

We observed that most models have no problem predicting repeated notes, the difficult part is predicting transitions. A good model thus must be able to successfully predict transitions. In order to evaluate this ability, we compute the cross entropy only on frames where there is a transition.

Let  $Tr$  be the subset of  $\llbracket 1, T - 1 \rrbracket$  such that:

$$t \in Tr \Leftrightarrow M_t \neq M_{t-1} \quad (3.6)$$

For each  $t \in Tr$ , we define  $d(t)$  the number of bins that differ between  $M_t$  and  $M_{t-1}$ , ie.  $d(t) = |M_t - M_{t-1}|_0$ . We define the transition cross entropy as:

$$\mathcal{H}_{tr} = \frac{1}{|Tr|} \sum_{t \in Tr} \frac{\mathcal{H}(M_t, \hat{M}_t)}{d(t)} \quad (3.7)$$

where  $|\cdot|$  denotes the cardinality. We divide by  $d(t)$  as we observe experimentally that  $\mathcal{H}(M_t, \hat{M}_t)$  is proportional to  $d(t)$ . Indeed, as the models have a tendency to repeat the previous input, each note that differs from the previous input will be an additional source of errors.

### Steady-state Cross entropy

The transition cross entropy evaluates the ability of an MLM in difficult cases. Still, we expect an MLM to perform also well in the simple cases, that is, when notes are held or repeated. We thus define the steady-state cross entropy as:

$$\mathcal{H}_{ss} = \frac{1}{T - 1 - |Tr|} \sum_{t \in \llbracket 1, T-1 \rrbracket \setminus Tr} \mathcal{H}(M_t, \hat{M}_t) \quad (3.8)$$

Here, we do not normalise by the number of active notes as it has no influence on the cross entropy value. What matters is the fact that the previous frame is repeated, not which notes are active in that frame.

### Pitch-profile Cross entropy

In Section 3.3, we observed that the network seems to be able to recognise the pitch distribution in certain cases, and appears to give higher probabilities to notes that are in-key. To evaluate quantitatively this ability, we introduce the pitch-profile cross entropy, which assesses how relevant to the piece the erroneous outputs of the system are.

In our case, we focus on whether erroneous outputs are in-key. It has to be noted that this is not necessarily the best way to assess whether a prediction

fits a piece. Indeed, in many instances, out-of-key notes are important to music pieces. For instance, in the introduction of *Für Elise* by L.W. Beethoven, the second pitch corresponds to the tritone, a highly dissonant and very unlikely pitch in tonal music, yet, it is one of the features that make this piece recognisable, and as such, should not be removed. We justify using this criterion by the fact that such events remain quite rare, and that in the majority of cases, removing out-of-key notes indeed tends to decrease the proportion of false positives when averaged over a dataset, but we acknowledge the fact that in some instances, deleting such rare events might result in important modifications of a music piece.

In our case, we define the scale of a piece as the set of MIDI pitches (not pitch classes) that are common in a piece. We consider that a pitch is in the scale of the piece if it is active for more than 5% of the duration (in seconds) of the example. This allows us to remove accidentals and ornaments. The threshold of 5% was set arbitrarily, and its influence is discussed in Section 3.4.3. Using such a definition of a scale has various advantages: we do not have to rely on potentially imperfect key annotations, it allows us to take into account potentially frequent out-of-key notes when unusual modes are used, and it takes into account the note range of a piece.

We use the times of the key signature changes in the Piano-midi.de dataset to define constant-pitch-profile regions. We then compute one scale per constant-pitch-profile region. Let  $(t_i)$  be the series of timesteps at which the key signature changes. We define the constant-pitch-profile regions as  $K_i = \llbracket t_i, t_{i+1} \rrbracket$ . Let  $N_{p,i}$  the subset of  $N$  such that:

$$N_{p,i} = \{(p_n, s_n, e_n) \in N \mid p_n = p \wedge (s_n \in K_i \vee e_n \in K_i)\} \quad (3.9)$$

We then define the pitch-profile of a piece  $P(t)$  for all  $t \in K_i$  as the subset of  $\llbracket 0, 87 \rrbracket$  such that pitch  $p$  is active for more than 5% of the duration of  $K_i$  ie.:

$$p \in P(t) \iff \frac{\sum_{N_{p,i}} \min(e_n, t_{i+1}) - \max(s_n, t_i)}{t_{i+1} - t_i} > 0.05 \quad (3.10)$$

We subsequently define a scale vector  $P(t)$  such that  $P(t)_p = 1 \iff p \in P(t)$ .  $P(t)$  is defined based on the durations of notes in seconds rather than in number of timesteps in order to compare more fairly different timesteps.

We only want to evaluate how close to the scale the erroneous predictions are. Indeed, our focus here is to measure to what extent the false positives, despite being mistakes, make sense from a musical point of view. In order to get rid of the influence of the correct notes, we do not consider in the computation of the

pitch-profile cross entropy the bins where the target is equal to 1. We call the ensemble of such bins  $B$ :

$$B = \{(p, t) \in \llbracket 0, 87 \rrbracket \times \llbracket 1, T - 1 \rrbracket \mid M[p, t] = 0\} \quad (3.11)$$

The pitch-profile cross entropy is then defined as the cross entropy between the false positive outputs and the scale, counting only the false positive bins. For a given bin  $(t, p)$ , the pitch-profile cross entropy is given as:

$$\mathcal{H}_{pp}(p, t) = -P(t)_p \log(\hat{M}[p, t]) - (1 - P(t)_p) \log(1 - \hat{M}[p, t]) \quad (3.12)$$

As the number of false positive bins changes from frame to frame, rather than defining it for each frame and then averaging it (which would give more weight to bins from frames where there are more notes), we define the pitch-profile cross entropy for a piano roll as follows:

$$\mathcal{H}_{pp} = \frac{1}{|B|} \sum_{(p,t) \in B} \mathcal{H}_{pp}(p, t) \quad (3.13)$$

### Transition-Pitch-profile Cross entropy

In order to investigate more specifically what happens on transitions, we define the transition-pitch-profile cross entropy. It is defined similarly as the pitch-profile cross entropy, but is only computed on transition frames. We call  $B_t$  the subset of pitches such that  $B_t = \{p \in \llbracket 0, 87 \rrbracket \mid (p, t) \in B\}$ . We then have:

$$\mathcal{H}_{pp,tr} = \frac{1}{\sum_{t \in Tr} |B_t|} \sum_{p \in B_t, t \in Tr} \mathcal{H}_{pp}(p, t) \quad (3.14)$$

### Steady-State-Pitch-profile Cross entropy

By analogy with  $\mathcal{H}_{ss}$  (see Section 3.4.2), we also define the steady-state-pitch-profile cross entropy:

$$\mathcal{H}_{pp,ss} = \frac{1}{\sum_{t \notin Tr} |B_t|} \sum_{p \in B_t, t \notin Tr} \mathcal{H}_{pp}(p, t) \quad (3.15)$$

## 3.4.3 Discussion on the proposed evaluation metrics

These metrics are imperfect, in the sense that they capture certain aspects of the performance of an MLM, but fail to capture others. In the same way that precision and recall give information on different aspects of a classification system, these metrics complement each other. A good model should thus perform

well in all, or most of them. In what follows, we describe situations in which a model might give good values, but actually be weak, or vice versa.

#### $\mathcal{H}_{tr}$ and $\mathcal{H}_{ss}$

$\mathcal{H}_{tr}$  only captures what happens on transitions. As a result, a model that is uncertain all the time might perform similarly to a model that is uncertain only at transition times. Conversely,  $\mathcal{H}_{ss}$  captures only what happens on steady-state frames. A model that always confidently predicts a note will be prolonged to the next timestep will have a good  $\mathcal{H}_{ss}$ , even though it might fail to predict transitions. A model should thus aim to score both good  $\mathcal{H}_{tr}$  and good  $\mathcal{H}_{ss}$ , which would indicate that it both knows when a note will be held, and successfully predicts the appearance of new notes.

Another potential issue is the fact that in some cases, there might be no steady-state frames. This happens in particular with *note-long* and *event* time-steps. In this case,  $\mathcal{H}_{ss}$  is ill-defined; we do not take into account such pieces when computing the average  $\mathcal{H}_{ss}$  on a dataset.

#### $\mathcal{H}_{pp}$ , $\mathcal{H}_{pp,tr}$ and $\mathcal{H}_{pp,ss}$

$\mathcal{H}_{pp}$ ,  $\mathcal{H}_{pp,tr}$  and  $\mathcal{H}_{pp,ss}$  all penalise in-key true negatives. For instance, an MLM that would always predict  $S(t)$  at every timestep, would score very good  $\mathcal{H}_{pp}$ , even though it fails to predict any specific note. This can be controlled with  $\mathcal{H}_{ss}$ , as a model that makes a lot of mistakes or fails to predict any note with confidence would score low on that metric.

One element that can influence these metrics is the threshold used to define  $P(t)$ . The higher the threshold, the more in-key false positives will be penalised, but true negatives will also be less penalised. Initial observations show that as the threshold increases,  $\mathcal{H}_{pp}$  gets better, meaning that the penalty given to in-key true negative outweighs the reward given to in-key false positives. In particular, this metric can only be used to compare models on the same dataset or on the same piece, but not to compare the performance of a single model on different pieces, as the number of pitches in the pitch profile is what has the most influence on  $\mathcal{H}_{pp}$ .

The second conclusion that can be drawn from this preliminary experiment is that the threshold chosen does not seem to have an influence on the relative ordering of the models, which means that any threshold can be chosen without influencing the conclusions of the study. We choose 5% rather than 0 because we still want to avoid including passing notes in the pitch profile as much as possible.

### 3.4.4 Combining metrics into one loss

All of the above metrics have their importance, and capture different aspects of the performance of an MLM. Still, it can be useful to try and combine them into one single value, in particular in order to use them as a loss function for model training. We thus propose the following formulation, with parameters  $\Theta = (w_{tr}, w_{ss}, \alpha)$ :

$$\mathcal{S}_\Theta = \sqrt{(w_{tr}\mathcal{H}_{tr} + w_{ss}\mathcal{H}_{ss})^{1+\alpha}(w_{tr}\mathcal{H}_{pp,tr} + w_{ss}\mathcal{H}_{pp,ss})^{1-\alpha}} \quad (3.16)$$

where  $(w_{tr}, w_{ss}) \in \mathbb{R}_+^2$  and  $\alpha \in [-1, 1]$ . When  $\Theta = (1, 1, 0)$ , we omit the  $\Theta$  subscript and we simply write it as  $\mathcal{S}$ .

Giving a higher value to  $w_{tr}$  or  $w_{ss}$  emphasises more transitions or steady states respectively. Giving a higher  $\alpha$  value emphasises more the  $\mathcal{H}$  factor compared to the  $\mathcal{H}_{pp}$  factor of  $\mathcal{S}_\Theta$ . All the summands are positive, provided that  $\hat{M}$  has values in  $[0, 1]$ . The lowest possible value is 0, reached when  $\hat{M}_t = M_t$  for all  $t$  (or when  $\hat{M}_t = P(t)$  for all  $t$ ).

By summing  $\mathcal{H}_{tr}$  and  $\mathcal{H}_{ss}$ , we take into account all the time-frequency bins, as in  $\mathcal{H}$ , but with a different weight. When  $w_{tr} = w_{ss}$ , the sum of steady-state frames and the sum of transition frames have the same weight overall, contrary to  $\mathcal{H}$ , where all frames have the same weight. As a result, when transitions are rare, they have a much higher weight with this metric than with  $\mathcal{H}$ , which helps put the emphasis on the less common case. When transitions are frequent (as is the case with *event* timesteps for instance), the steady-state frames end up having a higher weight than the transition frames. Similarly, summing  $\mathcal{H}_{pp,tr}$  and  $\mathcal{H}_{pp,ss}$  (with  $w_{tr} = w_{ss}$ ) allows us to evaluate all bins with respect to the pitch-profile, but gives more weight to the less common type of frames.

We choose the weighted geometric average to combine these two sums in order to avoid scaling problems and focus on relative differences in both terms. A geometric average allows us to keep the relative ordering invariant in  $\mathcal{S}$  as well. We decide to leave aside benchmark evaluation metrics when computing  $\mathcal{S}$ . Including  $\mathcal{H}$  would be redundant with  $\mathcal{H}_{tr} + \mathcal{H}_{ss}$ , as argued previously. As to  $\mathcal{F}$ , we leave it aside because we are here trying to evaluate probability distribution modelling rather than binary classification. Non-binary metrics are thus more appropriate than binary ones.

## 3.5 Experiments

Using this new set of metrics, along with the benchmark metrics, we carry out experiments to assess the influence of various parameters on the performance of

| Model           | $\mathcal{H}_{tr}$ | $\mathcal{H}_{ss}$ | $\mathcal{H}_{pp}$ | $\mathcal{H}_{pp,tr}$ | $\mathcal{H}_{pp,ss}$ | $\mathcal{S}$ |
|-----------------|--------------------|--------------------|--------------------|-----------------------|-----------------------|---------------|
| TIME-SHORT      |                    |                    |                    |                       |                       |               |
| <i>Baseline</i> | 6.95               | <b>0.088</b>       | 1.28               | 1.22                  | 1.28                  | 4.12          |
| LSTM            | <b>4.58</b>        | 0.164              | <b>1.26</b>        | <b>1.12</b>           | <b>1.26</b>           | <b>3.31</b>   |
| TIME-LONG       |                    |                    |                    |                       |                       |               |
| <i>Baseline</i> | 6.94               | <b>0.088</b>       | 1.22               | 1.19                  | 1.27                  | 4.09          |
| LSTM            | <b>2.66</b>        | 2.56               | <b>0.714</b>       | <b>0.691</b>          | <b>0.742</b>          | <b>2.67</b>   |
| NOTE-SHORT      |                    |                    |                    |                       |                       |               |
| <i>Baseline</i> | 6.94               | <b>0.088</b>       | 1.26               | 1.21                  | 1.28                  | 4.12          |
| LSTM            | <b>3.05</b>        | 0.524              | <b>1.07</b>        | <b>0.811</b>          | <b>1.15</b>           | <b>2.61</b>   |
| NOTE-LONG       |                    |                    |                    |                       |                       |               |
| <i>Baseline</i> | 6.94               | <b>0.088</b>       | 1.23               | 1.2                   | 1.26                  | 4.09          |
| LSTM            | <b>2.46</b>        | 1.83               | <b>0.856</b>       | <b>0.736</b>          | <b>0.995</b>          | <b>2.65</b>   |
| EVENT           |                    |                    |                    |                       |                       |               |
| <i>Baseline</i> | 6.94               | <b>0.088</b>       | 1.2                | 1.19                  | 1.28                  | 4.12          |
| LSTM            | <b>2.37</b>        | 4.14               | <b>0.658</b>       | <b>0.655</b>          | <b>0.702</b>          | <b>2.94</b>   |

Table 3.2: Prediction performance for various timesteps assessed with the proposed metrics. Bold values correspond to the best result for each timestep, and we underline it when it is significantly better (i.e.  $p < 0.05$  with a paired t-test).

our system. In Section 3.5.1, we first check whether the qualitative observations made in Section 3.3.3 are reflected by the new metrics. In Section 3.5.2, we compare our models against various models in the literature, trained with usual losses and the  $\mathcal{S}$  loss when applicable. In Section 3.5.3, we investigate what effect training our model using  $\mathcal{S}_\Theta$  as loss with various  $\Theta$  has on the predictions. Finally, in Section 3.5.4, we apply our MLMs to AMT, and investigate how the choice of parameters  $\Theta$  when training our MLMs with  $\mathcal{S}_\Theta$  influence AMT performance.

### 3.5.1 Influence of the time step

We re-evaluate the preliminary experiment using this new set of metrics to compare quantitatively how the model behaves in each of the five configurations: *time-short*, *time-long*, *note-short*, *note-long*, and *event*. Results can be found in Table 3.2, and plotted in Figure 3.3. We also include values for  $\mathcal{S}$  for completeness.

#### Comparison against naive baseline

Comparison with the naive baseline shows a similar trend when evaluated with the proposed metrics. It appears, as discussed in Section 3.3, that the longer

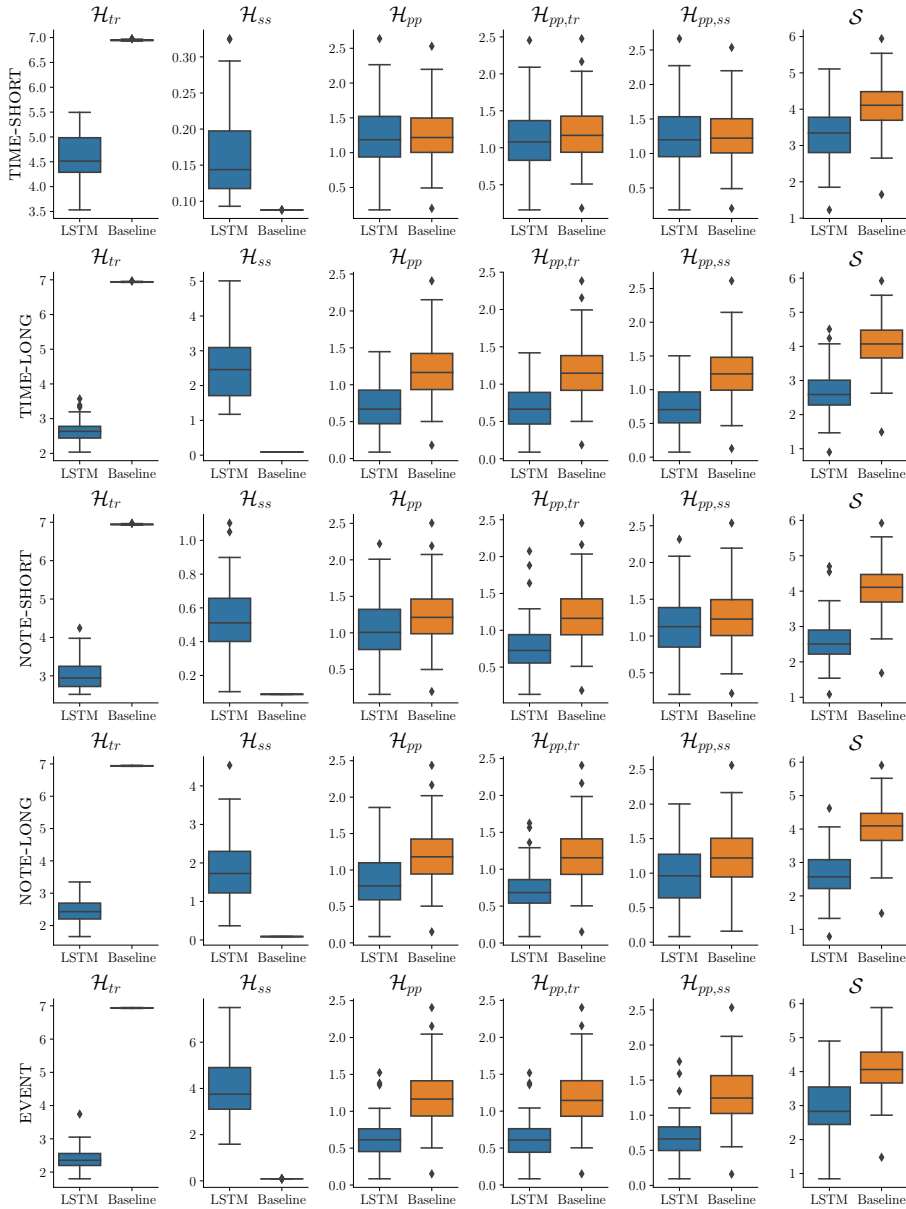


Figure 3.3: Comparison of the prediction performance for the LSTM and baseline models for various timesteps assessed with the proposed metrics. Columns correspond to metrics, rows correspond to timesteps.

the timestep, the more improvement we see over the naive baseline model for  $\mathcal{H}_{tr}$ ,  $\mathcal{H}_{pp}$ ,  $\mathcal{H}_{pp,tr}$  and  $\mathcal{H}_{pp,ss}$ . For  $\mathcal{H}_{ss}$ , the naive baseline is always better, which is understandable, as it is only wrong when there are transitions, not on steady states. Interestingly, we can see that in the *time-short* configuration,  $\mathcal{H}_{tr}$  is lower than that of the baseline model. This confirms the observation made in Section 3.3.3 that in some instances, despite using a very short timestep, the LSTM is able to adapt its behaviour when there are transitions, and is able to aggregate a temporal context long enough (at least a few notes, which depends on note durations and tempo) to predict when note changes might happen.

For the baseline model, we can also see that results across timesteps are very consistent, which was not the case for benchmark metrics. This suggests that comparison of results across timesteps might make more sense with the proposed metrics than with benchmark metrics, as a model behaving similarly gets similar scores.

### Time step length

The influence of the length of timesteps can be assessed by comparing *time-short* and *time-long* on one side, and *note-short* and *note-long* on the other. In both cases, the same trend can be observed: longer timesteps correspond to a bigger decrease in  $\mathcal{H}_{tr}$  compared to the baseline, but a bigger increase in  $\mathcal{H}_{ss}$ . This supports the idea that the shorter the timestep, the more confident the model is that the current note is going to be active at the next timestep.

We can also observe that both with time-based and note-based timesteps, the longer the timestep, the more improvement in  $\mathcal{H}_{pp}$ ,  $\mathcal{H}_{pp,tr}$  and  $\mathcal{H}_{pp,ss}$  there is compared to the baseline. This is explained by two factors: first, longer timesteps mean that the network can more easily aggregate a bigger temporal context, which can help recognise which notes are likely to occur. Moreover, as mentioned in Section 3.4.3, with shorter timesteps, the network tends to be more confident in continuations, predicting very few false positives. Since true negatives are penalised by these metrics, it drives them up.

### Time-based vs. Note-based

To compare the effect of using a tempo-normalised timestep, we can compare *time-long* and *note-long*. In both cases,  $\mathcal{H}_{tr}$  is quite low: the model has learned to give lower probabilities to the current note, which helps it being less confidently wrong on transitions, and consider other notes as likely. However, for *note-long*, there is a bigger improvement in  $\mathcal{H}_{ss}$  compared to Baseline than with *time-long*. In particular,  $\mathcal{H}_{ss}$  is closer to  $\mathcal{H}_{tr}$  for *time-long* than for *note-long*. Moreover, for *time-long*, the difference between  $\mathcal{H}_{pp,ss}$  and  $\mathcal{H}_{pp,tr}$  is lower than

for *note-long*. It shows that *time-long* tends to be quite uncertain everywhere, while *note-long* is able to adapt its behaviour on transitions, and moreover is able to adapt to the pitches of the current piece.

### Event timesteps

Comparing *event* timesteps with other timesteps is difficult, as they are defined in a quite different manner. That said, we can see that for *event* timesteps, improvement over the baseline is greater for  $\mathcal{H}_{tr}$  and  $\mathcal{H}_{ss}$  is worse. This indicates that the network has high uncertainty, including when notes are held, and favours transitions. The network is also able to match the pitch profile of the piece, given that  $\mathcal{H}_{pp}$ ,  $\mathcal{H}_{pp,tr}$  and  $\mathcal{H}_{pp,ss}$  are the lowest compared to baseline. This can also be explained by the presence of more false positives than with other timesteps, but still their distribution corresponds to that of the pieces.

### 3.5.2 Comparison with other models

We compare two versions of our model, one trained with  $\mathcal{H}$  and one with the  $\mathcal{S}$  loss, with the RNN-RBM [Boulanger-Lewandowski et al., 2012] and the diagonal RNN [Subakan and Smaragdis, 2017] (we will refer to it as diagRNN). For the latter, we use one single layer of diagonal LSTM units and early stopping for a fairer comparison. For the diagRNN, we also train two different versions: one trained with the  $\mathcal{H}$  loss, as recommended in [Subakan and Smaragdis, 2017], and one with the  $\mathcal{S}$  loss, to investigate the effect of the  $\mathcal{S}$  loss on various architectures. We cannot do so with the RNN-RBM, as it uses the free-energy loss rather than the cross entropy, and our custom loss cannot be trivially adapted in that case. We observed in previous experiments that note-based timesteps seemed to give better results (they are able to better predict transitions, and show a better balance between  $\mathcal{H}_{tr}$  and  $\mathcal{H}_{ss}$ ). We thus conduct all experiments with *note-long* timesteps, as well as *note-short* timesteps. Results are summarised in Table 3.3, plotted in Figure 3.4 and we show some example outputs with *note-long* timesteps in Figure 3.5. Outputs with *note-short* timesteps can be found on our supplementary materials webpage<sup>4</sup>.

In terms of  $\mathcal{H}$ , the best performing model is the LSTM trained with  $\mathcal{H}$  loss. This conflicts with the conclusions from [Subakan and Smaragdis, 2017], which found that their diagonal RNN achieved lower  $\mathcal{H}$  than regular LSTMs, although it does not invalidate them given that the setups and datasets are slightly different. With *note-long* timesteps, the two models are nearly equivalent. It also has to be noted that the diagRNN has fewer parameters than the regular LSTM (115k vs. 376k). Interestingly, the diagonal RNN scores lower  $\mathcal{H}_{pp}$  than our

<sup>4</sup><http://c4dm.eecs.qmul.ac.uk/ycart/taslp20.html>

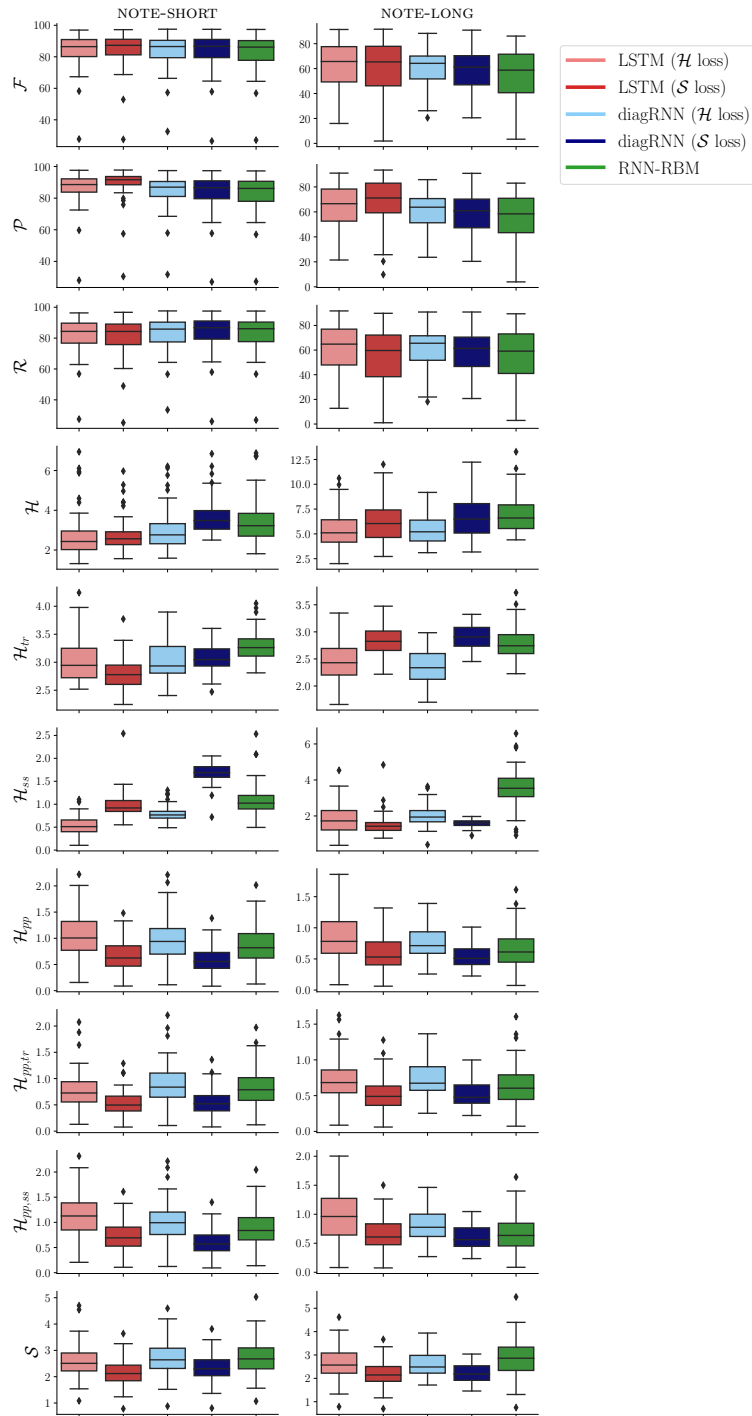


Figure 3.4: Comparison of the prediction performance of various models for various timesteps assessed with the benchmark and proposed metrics. Columns correspond to timesteps, rows correspond to metrics.

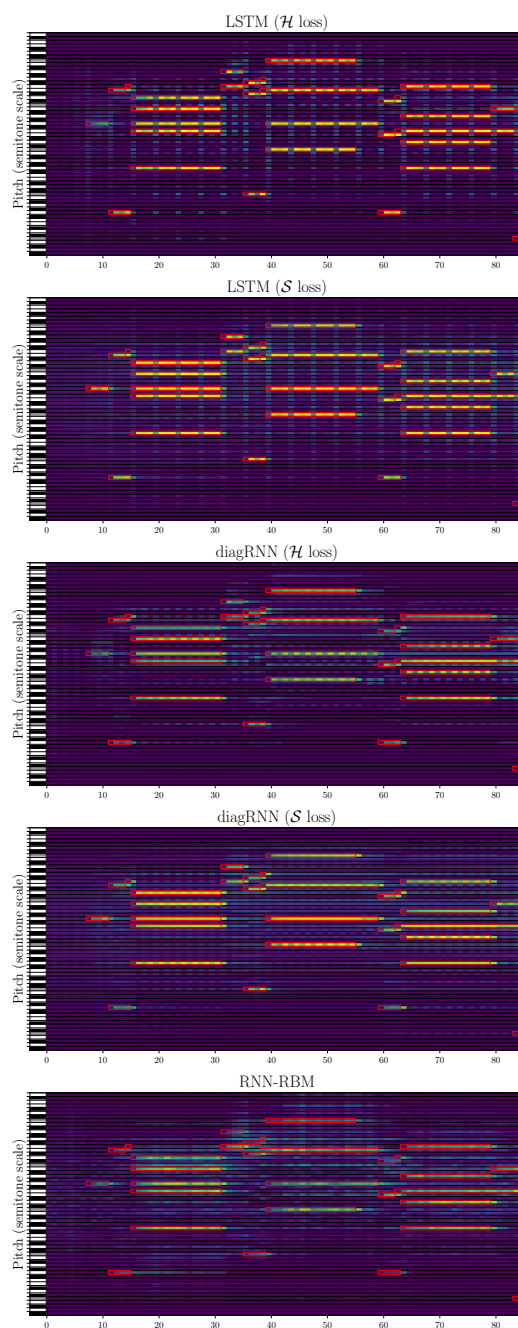


Figure 3.5: Comparison of sigmoid outputs for various models for the MIDI file `chpn-p7.mid` transposed into C, with *note-long* timesteps. The ground truth notes are overlaid as red rectangles. The notes of the scale correspond to the white keys on the left of each image. The tonic is in grey. The same color map is used across images.

| Note-short timesteps          |               |               |               |               |                    |                    |                    |                       |                       |               |
|-------------------------------|---------------|---------------|---------------|---------------|--------------------|--------------------|--------------------|-----------------------|-----------------------|---------------|
| Model                         | $\mathcal{F}$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{H}$ | $\mathcal{H}_{tr}$ | $\mathcal{H}_{ss}$ | $\mathcal{H}_{pp}$ | $\mathcal{H}_{pp,tr}$ | $\mathcal{H}_{pp,ss}$ | $\mathcal{S}$ |
| LSTM ( $\mathcal{H}$ loss)    | 84.1          | 86.3          | 82.0          | <b>2.76</b>   | 3.05               | <b>0.524</b>       | 1.07               | 0.811                 | 1.15                  | 2.61          |
| LSTM ( $\mathcal{S}$ loss)    | <b>84.7</b>   | <b>89.0</b>   | 81.0          | 2.79          | <b>2.81</b>        | 0.975              | 0.691              | <b>0.548</b>          | 0.737                 | <b>2.16</b>   |
| diagRNN ( $\mathcal{H}$ loss) | 83.9          | 84.5          | 83.3          | 3.03          | 3.04               | 0.79               | 1.0                | 0.918                 | 1.03                  | 2.68          |
| diagRNN ( $\mathcal{S}$ loss) | 83.7          | 83.7          | <b>83.7</b>   | 3.7           | 3.08               | 1.69               | <b>0.601</b>       | 0.565                 | <b>0.613</b>          | 2.33          |
| RNN-RBM                       | 83.1          | 83.4          | 82.9          | 3.49          | 3.3                | 1.12               | 0.889              | 0.85                  | 0.902                 | 2.74          |

| Note-long timesteps           |               |               |               |               |                    |                    |                    |                       |                       |               |
|-------------------------------|---------------|---------------|---------------|---------------|--------------------|--------------------|--------------------|-----------------------|-----------------------|---------------|
| Model                         | $\mathcal{F}$ | $\mathcal{P}$ | $\mathcal{R}$ | $\mathcal{H}$ | $\mathcal{H}_{tr}$ | $\mathcal{H}_{ss}$ | $\mathcal{H}_{pp}$ | $\mathcal{H}_{pp,tr}$ | $\mathcal{H}_{pp,ss}$ | $\mathcal{S}$ |
| LSTM ( $\mathcal{H}$ loss)    | <b>63.3</b>   | 64.8          | <b>62.3</b>   | <b>5.4</b>    | 2.46               | 1.83               | 0.856              | 0.736                 | 0.995                 | 2.65          |
| LSTM ( $\mathcal{S}$ loss)    | 60.6          | <b>67.7</b>   | 55.8          | 6.19          | 2.81               | <b>1.52</b>        | 0.59               | 0.532                 | 0.667                 | <b>2.22</b>   |
| diagRNN ( $\mathcal{H}$ loss) | 61.2          | 61.3          | 61.4          | 5.42          | <b>2.37</b>        | 2.04               | 0.775              | 0.735                 | 0.843                 | 2.6           |
| diagRNN ( $\mathcal{S}$ loss) | 58.6          | 58.6          | 58.6          | 6.66          | 2.91               | 1.59               | <b>0.555</b>       | <b>0.525</b>          | <b>0.612</b>          | 2.23          |
| RNN-RBM                       | 56.4          | 56.8          | 56.2          | 6.97          | 2.79               | 3.57               | 0.67               | 0.658                 | 0.69                  | 2.88          |

Table 3.3: Comparison of various MLMs, for note-short and note-long timesteps.

LSTM with both timesteps. However, they also both have high  $\mathcal{H}_{ss}$  compared to our LSTM, and lower difference between  $\mathcal{H}_{pp,tr}$  and  $\mathcal{H}_{pp,ss}$ , which shows they tend to be more uncertain.

The RNN-RBM on the other hand scores quite poorly in most metrics, except for  $\mathcal{H}_{pp}$ , where it outperforms the other  $\mathcal{H}$ -trained models. Its low performance can be attributed to the fact that it uses simple recurrent units instead of LSTM units. Besides, it does not use more recent optimisation methods such as Adam, relying instead on regular gradient descent. It also has very few parameters (56k), which makes it a very efficient model in terms of performance-complexity ratio.

Using the  $\mathcal{S}$  loss on the LSTM has different effects depending on the timestep. With *note-short*, since transitions are more rare, using the  $\mathcal{S}$  loss decreases  $\mathcal{H}_{tr}$ , while with *note-long*, it decreases  $\mathcal{H}_{ss}$ . In both cases,  $\mathcal{H}_{pp}$  also decreases. Interestingly, with *note-short*, training the model with  $\mathcal{S}$  does not lead to a degradation in terms of  $\mathcal{H}$ , and it leads to an increase in  $\mathcal{F}$ . We attribute this to the fact that using our loss encourages the network to pay attention to note transitions, rather than favouring note repetition. However, this conclusion does not transfer to *note-long*, where the original data contains more transitions. With the diagRNN, the  $\mathcal{S}$  loss has a slightly different effect: it mostly decreases  $\mathcal{H}_{pp}$  with both timesteps and  $\mathcal{H}_{ss}$  with *note-long*, but does not improve  $\mathcal{H}_{tr}$  with *note-short*.

Looking at the outputs of the systems, it appears that all other models fail to predict beat positions in the same way the LSTM does. The diagRNN does predict some alternating patterns (both with *note-long* and *note-short* timesteps), but does not seem to be able to predict patterns with a longer

period (e.g. 4 or 6 timesteps). The RNN-RBM fails to predict any rhythmic pattern, making very irregular predictions, probably due to the randomness in the Gibbs sampling process used to obtain these values. Still, they all manage to predict notes of the key with relative success, which is coherent with the quite low  $\mathcal{H}_{pp}$  values. Moreover, both the LSTM and the diagRNN with the  $\mathcal{S}$  loss do predict more in-key false positives than their  $\mathcal{H}$  counterparts, which is coherent with the previous observations. Besides, the diagRNN with the  $\mathcal{S}$  loss and *note-short* steps does not predict any transitions, while transitions are enhanced with the LSTM, which suggests that the model is simply not capable of learning these kinds of temporal patterns.

### 3.5.3 Influence of $\Theta$ with $\mathcal{S}_\Theta$ loss

In this section, we investigate the use of  $\mathcal{S}_\Theta$  as training loss for our system. In particular, we look into the effect of the  $\Theta$  parameters on model performance and perform a grid search with the following parameters:  $(w_{tr}, w_{ss}) \in \{(4, 1), (2, 1), (1, 1), (1, 2), (1, 4)\}, \alpha \in \{-1, -0.5, -0.25, 0, 0.25, 0.5, 1\}$ . For the sake of conciseness, we do not present these experiments with all timesteps, we show results only with *note-long* timesteps. We ran the same grid search with other timesteps, and similar conclusions can be drawn. Results of the grid search for *note-long* timesteps can be found in Figure 3.6.

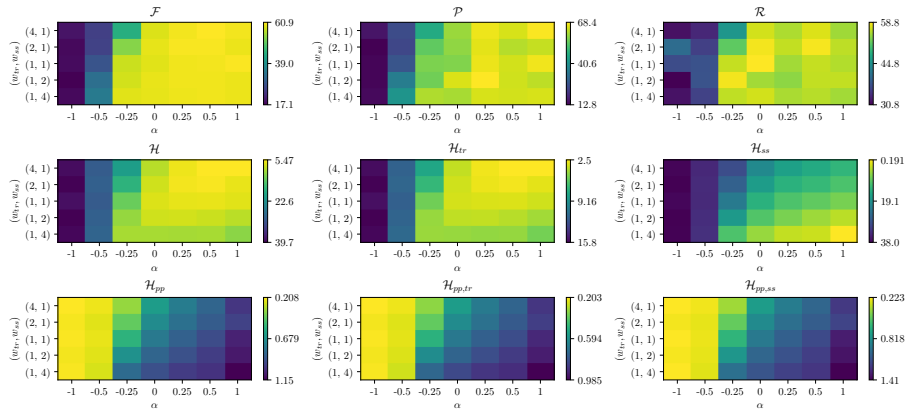


Figure 3.6: Evaluation of various MLMs trained with  $\mathcal{S}_\Theta$  loss, with various  $\Theta$  configurations, all with *note-long* timesteps. Each grid corresponds to a metric defined in Section 3.4, rows correspond to different  $(w_{tr}, w_{ss})$  configurations, and columns correspond to different  $\alpha$ . Brighter colours correspond to better performance (higher for  $\mathcal{F}$ ,  $\mathcal{P}$  and  $\mathcal{R}$ , lower for cross-entropy-based metrics).

Inspecting the results of our MLMs with various  $\Theta$  configurations, we can see that the parameters influence the end result as expected. When increasing the value of  $\alpha$ ,  $\mathcal{H}$  gets better, while decreasing it improves  $\mathcal{H}_{pp}$ . When  $w_{tr}$  is

higher than  $w_{ss}$ ,  $\mathcal{H}_{tr}$  improves, and conversely,  $\mathcal{H}_{ss}$  improves when  $w_{ss}$  is higher than  $w_{tr}$ . We can also see that using a higher  $\alpha$  improves  $\mathcal{F}$ . This makes sense as a higher  $\alpha$  will make the network try to predict specific notes, while a lower  $\alpha$  will make the network always predict notes of the pitch profile, failing to predict any specific note.

Some results are more unexpected. We can see that  $\mathcal{H}_{pp,tr}$  and  $\mathcal{H}_{pp,ss}$  behave exactly the same. They have different values, but they are affected similarly by  $\Theta$ . We infer that this is due to the fact that the target for  $\mathcal{H}_{pp,tr}$  and  $\mathcal{H}_{pp,ss}$  is the same, it does not depend on whether the frame is a transition or not. Therefore, the model tends to behave similarly on transition frames and steady-state frames with respect to these metrics, in particular when  $\alpha$  is low. Besides, we see that  $\mathcal{H}_{pp,ss}$  improves when  $w_{tr}$  has a higher weight, which is somewhat counter-intuitive. Our interpretation is that it is related to the fact that a high  $w_{tr}$  encourages MLM false positives. As the false positives tend to overall have higher activations, and they usually correspond to notes of the pitch profiles,  $\mathcal{H}_{pp}$  gets lower, since in-key true negatives are penalised by  $\mathcal{H}_{pp}$ .

When inspecting outputs, we can observe that models with a lower  $\alpha$  tend to have more false positive activations. Models trained with higher  $w_{tr}$  also have stronger false positive activations, as transitions are encouraged. This improvement is particularly visible with *time-short* timesteps: when trained with  $\mathcal{S}$ , the model is able to infer notes of the scale, while it predicts nearly no false positives when trained with  $\mathcal{H}$ . In Figure 3.7, an example with *time-short* timesteps is shown. Here, it can be seen that when trained with  $\mathcal{S}$ , the model is able to predict some note changes at specific timesteps, materialised as vertical lines on the figure. It has to be noted that this is particularly visible in this specific example as notes are short and fairly regular, which makes note changes easier to predict; this is not as visible on examples with longer notes.

Upon closer inspection, we can see that the model both reacts to transitions, and predicts them to some extent. On the one hand, when there is a note change, the model deactivates all non-active notes at the following timestep, as it is quite rare that another note change happens right after. This corresponds to the visible vertical lines, and corresponds to a reaction after a transition. On the other hand, the model also gradually activates likely notes when a note change might happen. This can be seen as a blur before transitions, and corresponds to predictions. None of this was the case when simply training with  $\mathcal{H}$ , showing the effectiveness of the new designed loss in making models more musical when transitions are rare.

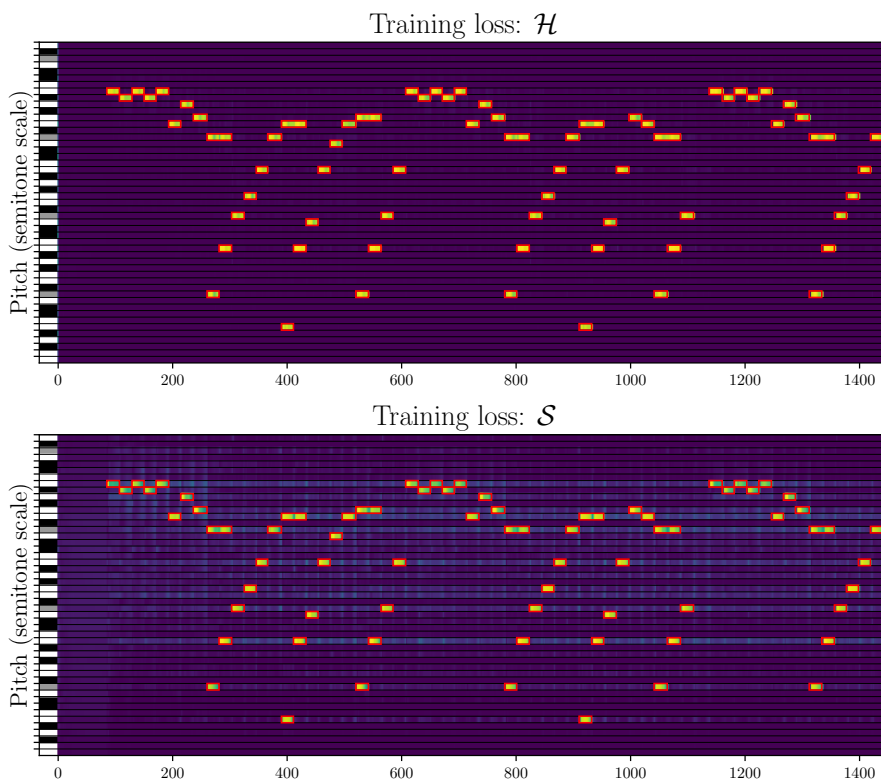


Figure 3.7: Comparison of sigmoid outputs for various training losses, all with time-short timesteps for the MIDI file `elise.mid`. The ground truth notes are overlaid as red rectangles. The notes of the scale correspond to the white keys on the left of each image (harmonic minor mode). The tonic is in grey. The same color map is used across images.

### 3.5.4 Application to AMT

#### Description

In this section, we investigate the impact of our new metrics and losses on the task of AMT. To do so, we use exactly the same setup as described in Chapter 5. In other words, we post-process the output of the convolutional neural network (CNN) acoustic model described in [Kelz et al., 2016] (see also Section 2.3.2). At each timestep, we use our LSTM to make predictions for the next timestep based on the previous binary outputs of the system. We define the distribution of predictions for the next timestep as the weighted sum of the predictions of the LSTM and the acoustic model, with constant weights 0.2 and 0.8 respectively (denoted as CW in Section 5.3.3). In other words, let  $P[p, t]$  be the output of the acoustic model at time  $t$  for pitch  $p$ , and  $\hat{M}[p, t]$  the predictions made by the MLM for the same time and pitch, the combined likelihood  $O[p, t]$  of pitch

$p$  being active at timestep  $t$  is given by:

$$O[p, t] = 0.8 * P[p, t] + 0.2 * \hat{M}[p, t] \quad (3.17)$$

Outputs for the next timestep are obtained by sampling the most likely 88-dimensional binary vectors from the combined distribution  $O_t = (O[p, t])_{0 \leq p < N_p}$  (see Section 5.2.4 for details on the sampling procedure). The best sequence overall is then obtained with Viterbi decoding with beam search (see Section 5.2.4 for details).

As in Chapter 5, we run our experiments with 16th-note timesteps and 40ms timesteps. We also ran similar experiments with 48th-note timesteps and obtained similar conclusions (omitted for brevity). Similarly to Chapter 5, regardless of the timestep, all results are converted back to 40ms timesteps before being compared to the target. We use the same evaluation metrics as described in Section 5.3.4, namely framewise metrics ( $P_f$ ,  $R_f$  and  $F_f$ ) and onset-only notewise metrics ( $P_{n,On}$ ,  $R_{n,On}$ , and  $F_{n,On}$ ). MLMs are trained using the setup described in the current chapter, but use the same dataset splits as Chapter 5. We choose to use the CW configuration to combine the predictions of the MLM and the acoustic model in order to use a simpler, more standard approach.

We investigate how the choice of  $\Theta$  when using  $\mathcal{S}_\Theta$  as MLM training loss influences AMT performance. Using the same grid search parameters as in Section 3.5.3, we evaluate the AMT performance with each MLM. We also report, for both timesteps, results of a simple baseline that consists in thresholding the acoustic model predictions at 0.5. We do not apply any smoothing on the outputs, as we are mostly interested here in the comparison between the various MLM losses rather than overall best performance. We show the full results of the grid search with 16th note timesteps in Fig. 3.8 (the results for other timesteps are available on our supplementary materials webpage<sup>5</sup>). AMT results for the baseline, cross-entropy-trained-MLM decoding, and best-performing MLM are shown in Table 3.4, along with the  $\Theta$  parameters that yield best performance.

## Results

When considering the effect of  $\Theta$  on AMT performance, we can observe similar tendencies for all tested timesteps. Using a lower  $\alpha$  results in a higher  $\mathcal{R}$ , both framewise and notewise, as the MLM tends to predict all the notes in the pitch profile, at every timestep. Conversely, a higher  $\alpha$  results in a higher  $\mathcal{P}$ , as the MLM focuses more on predicting fewer notes, and only the very likely ones. It also appears that with higher  $w_{tr}$ , framewise  $\mathcal{P}$  increases, while a

<sup>5</sup><http://c4dm.eecs.qmul.ac.uk/ycart/tas1p20.html>

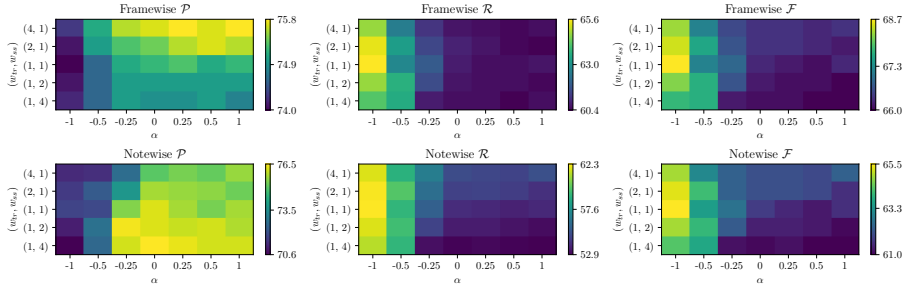


Figure 3.8: AMT performance of various MLMs trained with the  $\mathcal{S}_\Theta$  loss, with various  $\Theta$  configurations. Each grid corresponds to a metric, rows correspond to different  $(w_{tr}, w_{ss})$  configurations, columns correspond to different  $\alpha$ . Brighter colours correspond to better values.

higher  $w_{ss}$  increases notewise  $\mathcal{P}$ . When  $w_{ss}$  is high, very few false positives are predicted by the MLM, which increases notewise  $\mathcal{P}$ . On the other hand, when  $w_{tr}$  is high, some false positive notes might be detected, but since transitions are encouraged, they are not held, which improves framewise  $\mathcal{P}$ . While for  $\mathcal{P}$  and  $\mathcal{R}$ , these tendencies are fairly consistent across tested timesteps, results are more difficult to interpret for  $\mathcal{F}$ . No general tendency can be observed, and the best-performing  $\Theta$  configuration is quite different in each case.

Overall, we manage to get an improvement both in framewise and notewise  $\mathcal{F}$  with  $\mathcal{S}_\Theta$  compared to  $\mathcal{H}$  with all timesteps (although similarly to results reported in Section 5.4, notewise  $\mathcal{F}$  is below baseline for 16th note timesteps). This is a promising result, showing that by selecting appropriate  $\Theta$  parameters according to the task to be addressed, an improvement can be obtained without increasing model complexity. Moreover, consistent behaviour in terms of  $\mathcal{P}$  and  $\mathcal{R}$  can be observed across timesteps depending on the tuning of the  $\Theta$  parameters, which can motivate trying certain parameters in priority to enhance certain aspects of the system. For instance, with 40ms timesteps, baseline results have high notewise  $\mathcal{R}$  and low notewise  $\mathcal{P}$ , so it makes sense to use high  $w_{tr}$  and  $\alpha > 0$  to improve notewise  $\mathcal{F}$ . It is also interesting to note that in both cases, the MLM that yields best results is not the best-performing MLM according to  $\mathcal{H}$ : using  $\mathcal{H}$  as a way to evaluate MLMs is thus not a good reflection of their ability to perform well in extrinsic tasks, at least for AMT.

### 3.6 Conclusions

In this study, we investigated the performance of an LSTM for polyphonic music sequence modelling using various timesteps. We proposed new metrics to evaluate MLMs, and proposed a new, parametric loss that allows control over

| Model                                   | $P_f$       | $R_f$       | $F_f$       | $P_{n,On}$  | $R_{n,On}$  | $F_{n,On}$  |
|---|-------------|-------------|-------------|-------------|-------------|-------------|
| 40MS TIMESTEPS                          |             |             |             |             |             |             |
| Baseline                                | 72.6        | <b>65.0</b> | <b>67.8</b> | 51.1        | <b>67.5</b> | 56.8        |
| LSTM ( $\mathcal{H}$ loss)              | <b>73.3</b> | 64.1        | 67.6        | 60.6        | 64.1        | 61.2        |
| LSTM ( $\mathcal{S}_{(1,4,0,25)}$ loss) | 73.0        | 64.4        | 67.7        | <b>61.4</b> | 64.0        | <b>61.6</b> |
| 16TH NOTE TIMESTEPS                     |             |             |             |             |             |             |
| Baseline                                | 74.8        | 65.0        | 68.6        | 71.0        | <b>63.8</b> | <b>66.1</b> |
| LSTM ( $\mathcal{H}$ loss)              | <b>76.1</b> | 60.6        | 66.5        | <b>75.7</b> | 55.4        | 62.6        |
| LSTM ( $\mathcal{S}_{(1,1,-1)}$ loss)   | 74.0        | <b>65.6</b> | <b>68.7</b> | 71.8        | 62.3        | 65.5        |

Table 3.4: Comparison of AMT performance for various MLMs. All results are obtained using the CW configuration, as described in Section 5.3.3. The best values for each timestep are in bold.

the behaviour of the model. We showed that our model compared favourably with other MLMs in the literature, both quantitatively and qualitatively. We investigated how loss parameters influence model performance, both intrinsically and with AMT as an extrinsic task. We showed that the parameters influenced the precision and recall of AMT systems in consistent patterns, and that this could be exploited to obtain better transcription performance, without changing model complexity. In particular, we found no relationship between the MLM’s performance in terms of cross-entropy, and the AMT performance of the system, highlighting the need for new training losses and evaluation metrics. Besides, the MLMs that achieved best AMT performance were very different for different timesteps, showing the importance of such loss being parametrised and tuneable for each specific task.

The formulation of the new loss we proposed here is just one of the many possible ways to define it. We could have used a different way to combine all the metrics (*e.g.* arithmetic or harmonic mean of the  $\mathcal{H}$  and  $\mathcal{H}_{pp}$  components, multiplying the  $\mathcal{H}_{tr}$  and  $\mathcal{H}_{ss}$  instead of summing them), and used different ways to weight the various components. We settled for this one because it was the most straightforward in our opinion, but we encourage readers to experiment and fit the loss definition to their needs. To validate the usefulness of this loss in a broader music modelling context, more experiments would be required with other extrinsic tasks, for instance with music generation, or symbolic music classification. It would also be interesting to investigate the effect of this loss with more sophisticated architectures, such as for instance the Transformer model [Vaswani et al., 2017].

While our proposed metrics allow us to get more insight on how the model behaves empirically, they do not describe anything about the inner workings of the LSTM. Neural network interpretability is a difficult and open research

problem [Guidotti et al., 2018], that is beyond the scope of this thesis. Still, it would be interesting to look into the trained weights, or inspect the contents of the cell state with various kinds of input data, to obtain insights into how this behaviour is encoded into the model.

Finally, this whole study was made with the assumption that music is represented as a binary piano roll. However, this representation has its limitations, for instance the tradeoff between fine temporal resolution and compact representation, and not being able to differentiate between held and repeated notes. To address the latter, we could use other representations, for instance some piano rolls using more than 2 symbols (*e.g.* onset, continuation, and silence). The loss defined here could still be applied for other frame-based representations, and it would be interesting to see how it impacts predictions. We could also use representations that are not frame-based, but note-based, closer to the MIDI format, where note events are represented by their pitch, onset and offset. Then, our proposed loss could not be trivially adapted. Still, investigating different kinds of representations is an important question in exploring polyphonic music sequence prediction.

## Chapter 4

# Polyphonic Music Sequence Transduction

### 4.1 Introduction

As explained in Chapter 2, a common intermediate step in an AMT system is to obtain a posterioqram, and then post-process it to obtain a binary piano roll (typically, by thresholding it). In this section, we investigate neural network models to learn a mapping from a posterioqram to a piano roll.

This approach is different from MLM decoding, that we tackle in Chapter 5. Here the network is trained on aligned pairs of posterioqrams and binary piano rolls. The network learns to correct the mistakes made by a specific acoustic model, and can a priori not be used with other acoustic models. It can be related to image denoising [Fan et al., 2019], in the sense that a transduction system essentially tries to attenuate false positives and enhance false negatives. In what follows, we investigate transduction methods for AMT.

First, motivated by experiments in Chapter 3 and previous results [Korzeniowski and Widmer, 2017], we investigate whether using a musically-relevant timestep (in our case, 16th notes) can improve the transduction process, compared to time-constant timesteps (in our case, 10ms). Indeed, we have seen in Section 3.3 that using a musically-relevant timestep can help the model to learn the temporal dependencies in note sequences, in particular the fact that notes are likely to change on strong metrical positions. Moreover, using longer timesteps allows the model to take into account a longer past, which can help it model concepts such as tonality. These experiments are presented in Section 4.2. We find that although using 16th note timesteps does yield better results, this improvement comes mostly from the fact that durations are quantised, not

from better temporal modelling.

In another set of experiments, presented in Section 4.3, we investigate various design choices on the model itself. We compare two network architectures, and investigate the use of binary neurons (see Section 2.8.5) for the task of polyphonic sequence transduction, all evaluated on two different acoustic models. Overall, the best performing architecture is a CNN trained with F-measure loss with binary output neurons. We then run an experiment to compare this architecture with 16th note and time-constant steps, and find that with this model, 16th note steps improve results further than simply quantising the outputs, which suggests that this model is able to better model temporal dependencies.

In all the following experiments, we use the MAPS dataset [Emiya et al., 2010], which contains MIDI files of polyphonic piano music, along with aligned audio renditions, generated using synthetic pianos and Disklavier acoustic pianos (see Section 2.7.2 for details). Rhythm annotations for this dataset are available in the A-MAPS annotations (see Appendix A). We use these annotations in order to run our experiments using a timestep of a 16th note. In a real-life setting, such annotations are not available, and would have to be obtained using beat tracking algorithms. We consider them given in the following experiments as a proof of concept, in order to investigate what improvement can be obtained in an ideal case.

In what follows, we begin by investigating in Section 4.2 the use of musically-relevant timesteps for transduction. In Section 4.3, we compare various neural network architectures to perform that task, and look into the performance of the best model depending on the timestep used. Finally, in Section 4.4, we take a step back and reflect on the results we obtained.

The results in Section 4.2 were presented at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018), reference (5) from Section 1.4. The experiments in Section 4.3 were partly presented at the International Society for Music Information Retrieval Conference (ISMIR 2019), reference (9) from Section 1.4.

## 4.2 Investigating musically-relevant timesteps

### 4.2.1 Motivation and objectives

In this first experiment, we aim to investigate how using musically-relevant timesteps instead of shorter, time-constant timesteps impacts transduction performance. We have shown in Section 3.3 that in a music prediction setup, the size of the timestep has a large influence on the performance of the system. Since sequence transduction might also rely on the temporal dependencies

within the input sequence, we assume that using a musically-relevant timestep might improve the performance of the system. To test that hypothesis, and in accordance with experiments conducted in Chapter 3, we train two versions of a simple transduction model, one operating on 10ms timesteps (the timestep of the acoustic model used), and one on 16th note timesteps (a musically-relevant time unit, shown to give good results in Chapter 3), and we then compare their performance. Since we use ground-truth 16th note annotations, this is only a proof-of-concept experiment, to investigate how much improvement can be obtained in an ideal case.

In this section, we present the model we use in the experiments in Section 4.2.2. We describe the evaluation metrics in Section 4.2.3. We present the results of the experiments in Section 4.2.4. Finally, in Section 4.2.5, we discuss the results and propose some perspectives for future developments.

### 4.2.2 Models

In this section, we describe the multi-pitch detection and transduction models. We use the transduction model in two configurations: using timesteps of 10ms (that we call *time-based steps*), and using timesteps of a sixteenth note (that we call *note-based steps*).

#### Acoustic Model

To obtain the posteriors, we use the multi-pitch detection system of [Benetos and Weyde, 2015], which is based on Probabilistic Latent Component Analysis (PLCA). The system decomposes an input spectrogram into several probability distributions for pitch activations, instrument source contributions and tuning deviations, using a fixed dictionary of pre-extracted spectral templates. For this experiment, a piano-specific system is used, trained using isolated notes from the MAPS database [Emiya et al., 2010]. The output of the acoustic model is a real-valued matrix  $P$  of size  $N_p \times T$  with  $N_p = 88$ , each of the 88 rows corresponding to activations of one of the 88 keys of a piano over time, with a timestep of 10ms.

In the case of note-based timesteps, we have to downsample these posteriors, in order to get one value per sixteenth note step (we remind the reader that in this study, we consider the locations of the sixteenth note marks given). Formally, we have to transform  $P[p, t]$  into  $Q[p, s]$ , where  $t$  is a time index and  $s$  is a sixteenth-note step index. To do so, we average the posterior values corresponding to one timestep. In other words, given  $t_1$  and  $t_2$  the timesteps corresponding to  $s$  and  $s + 1$  respectively, we have:

$$Q[p, s] = \frac{\sum_{t=t_1}^{t_2} P[p, t]}{t_2 - t_1} \quad (4.1)$$

### Transduction Model

The goal of this study is to demonstrate how using musically-relevant timesteps can improve the performance of a multi-pitch detection system. For simplicity, and to limit interference with other techniques, we deliberately use a simple LSTM architecture for the transduction model. In particular, we choose not to use multiple layers, nor to use dropout [Hinton et al., 2012a] or any other regularisation method during training.

We thus use an LSTM with 88 inputs, one single hidden layer with  $N_h$  hidden nodes, and 88 outputs, one for each piano key, which are sent through a sigmoid function. The network is trained using the Adam optimiser [Kingma and Ba, 2015] with learning rate  $l$ , using the cross-entropy between the output of the sigmoid and the ground truth as cost function. We use as training loss the binary cross-entropy (CE) loss averaged over all entries  $\hat{M}[p, t]$  in the non-binary predicted piano roll  $\hat{M}$  as loss for the transducer model:

$$CE = \frac{1}{TN_p} \sum_{p=1}^{N_p} \sum_{t=1}^T M[p, t] \log \hat{M}[p, t] + (1 - M[p, t]) \log(1 - \hat{M}[p, t]) \quad (4.2)$$

where  $M[p, t] \in \{0, 1\}$  is the ground truth piano roll, indicating whether a note at timestep  $t \in \llbracket 1, T \rrbracket$  with pitch index  $p \in \llbracket 1, N_p \rrbracket$  is active. We use four sets of hyper-parameters, as a simpler alternative to extensive grid search:  $N_h \in \{128, 256\}$  and  $l \in \{0.001, 0.01\}$ .

The output of the network is then thresholded to obtain a binary piano roll. The threshold is determined by choosing the one that gives the best results on the validation dataset (see Section 4.2.4 for more information). An example comparing the input posterigram, the thresholded output of the LSTM and the ground truth is available in Figure 4.1.

### 4.2.3 Evaluation metrics

We evaluate our model with benchmark metrics as described in Section 2.6.2. We use only framewise ( $F_f$ ,  $P_f$ ,  $R_f$ ) and onset-only metrics ( $F_{n,On}$ ,  $P_{n,On}$ ,  $R_{n,On}$ ). We do not take the offsets into account as they are difficult to estimate properly with percussive sounds such as piano notes.

We compare two settings: *time-based* and *note-based*. In the time-based setting, computations are made using the time-based model. The results are

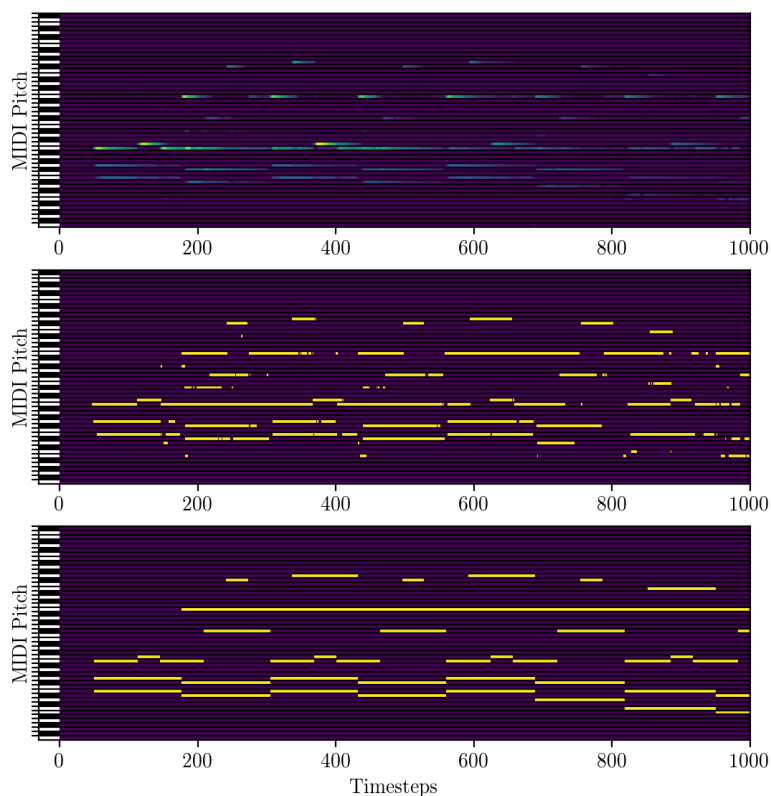


Figure 4.1: An example of input posterioqram (top) and thresholded output (middle) of the LSTM, compared to the ground truth (bottom). Computations were made with 10ms timesteps.

directly compared to the ground truth. In the note-based setting, computations are made using the note-based model. To compare them to the ground truth, we thus up-sample the output of the network back to 10ms timesteps, and use the same evaluation metrics.

Since our model does not output onsets and offsets explicitly, we treat any output 1 not preceded by a 1 as an onset, and any 0 not preceded by a 0 as an offset. We treat the ground truth the same after first converting it into a piano roll. Thus, our “notewise” metrics do not correspond with notes exactly, but rather as close as our output format can get.

The reason we apply this processing on the target as well is to have a fairer comparison, especially when using 16th note timesteps. Indeed, to be able to represent a repeated note, we need to have at least 1 timestep of silence between

two notes. With a 16th note timestep, this means that we cannot represent two repeated sixteenth note played one after the other. Repeated notes are much more likely to be merged, so we compare them against a similar target to get a better representation of the quality of the system. In any case, results are expected to be similar in most cases: when comparing the original list of notes and the list of notes obtained by post-processing the target piano roll, we obtain an  $F_{n,O_n}$  of 95% with 10ms timesteps, and 92% with 40ms timesteps. The two targets are thus very similar, so we assume that this comparison still is an accurate indication of the quality of a system.

#### 4.2.4 Experiments

##### Setup

We train our transduction model using the posterioqram output of the acoustic model as input, cut in 30-second chunks. We train two different networks: one operating on inputs with time-based timesteps, one on inputs with note-based timesteps. Both networks are trained for 100 epochs. The evaluation is only performed on the 30 first seconds of each file in the test set, as is typically done in related work.

As mentioned in Section 4.1, we use the MAPS dataset [Emiya et al., 2010], and the A-MAPS rhythm annotations (see Appendix A). We evaluate our system using 4-fold cross-validation, using the folds referred to as *Configuration 1* in [Sigtia et al., 2016]. Those folds were built to have no overlap in terms of music pieces between training and testing sets, but the piano models used can be found in both sets. As validation set, we randomly pick and set aside 15% of the training 30-second chunks. The acoustic model is trained using the same folds as the transduction model.

We compare our model against: median filter & thresholding and HMM smoothing [Poliner and Ellis, 2006] (see Section 2.3.3). We call the former *Baseline* and the latter *HMM*. The HMM method models each pitch by an on/off HMM, and then decodes the most likely sequence of hidden states using the Viterbi algorithm [Viterbi, 1967]. The initial probabilities and transition parameters are computed from ground truth annotations, one set of parameters per pitch. For the HMM, since the posterioqram activations are not bounded between 0 and 1 (they have values in  $[0, +\infty[)$ , they cannot be considered as observation probabilities. We thus model them as observations from an exponential distribution of parameter  $\lambda$  ( $\lambda > 0$ ), whose probability density function is defined as:

$$\Lambda(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (4.3)$$

We define a probability distribution  $p_{on}(P[p, t])$  for the hidden state “on”, with parameter  $\lambda$ , using the same parameters for all pitches. The probability distribution for the hidden state “off” is then defined as  $1 - p_{on}(P[p, t])$ . The  $\lambda$  parameter is chosen to optimise  $F_{n,On}$ . In both above cases, model parameters were estimated on the MAPS training folds.

To obtain note-based results from those systems, we downsample their binary outputs by activating a note for the considered timestep if it is active for more than 5% of the corresponding sixteenth note time interval, or for more than 2 frames. We choose this criterion because of the imprecision of the alignment: sometimes, the true onset of a note occurs slightly before the time indicated in the correspondence table, which we do not want to result in a note shifted by a whole sixteenth note.

## Results

| Time-based setting |             |             |             |             |             |             |
|--------------------|-------------|-------------|-------------|-------------|-------------|-------------|
| System             | $F_f$       | $P_f$       | $R_f$       | $F_{n,On}$  | $P_{n,On}$  | $R_{n,On}$  |
| Baseline           | 63.8        | 71.0        | 61.6        | <b>65.3</b> | 63.2        | <b>70.6</b> |
| HMM                | 55.2        | <b>74.1</b> | 48.1        | 61.8        | <b>86.2</b> | 50.9        |
| LSTM               | <b>66.3</b> | 67.0        | <b>67.8</b> | 57.2        | 51.1        | 69.3        |

| Note-based setting |             |             |             |             |             |             |
|--------------------|-------------|-------------|-------------|-------------|-------------|-------------|
| System             | $F_f$       | $P_f$       | $R_f$       | $F_{n,On}$  | $P_{n,On}$  | $R_{n,On}$  |
| Baseline           | 65.2        | 64.8        | 69.9        | <b>66.3</b> | 66.6        | <b>67.7</b> |
| HMM                | 56.3        | <b>70.5</b> | 51.4        | 58.5        | <b>81.9</b> | 48.0        |
| LSTM               | <b>67.1</b> | 65.9        | <b>71.0</b> | 62.2        | 59.6        | 67.0        |

Table 4.1: Benchmark evaluation metrics for all systems, evaluated on the MAPS subsets ENSTDkCl and ENSTDkAm, with best values in bold.

The results, averaged across 4 folds, are reported in Table 4.1. It turns out that in the vast majority of the experiments, the best performing configuration for the LSTM overall was 128 hidden nodes with a learning rate of 0.01; we only report the results for these parameters.

When evaluated on a frame basis, the LSTM is the best performing architecture. The improvement is particularly significant with time-based time-steps. The LSTM gets better results in the note-based setting compared to the time-based setting, which corroborates the idea that using a musically-relevant timestep improves performance.

On the other hand, with note metrics, the baseline model outperforms our system. Upon inspection of the output of the LSTM, it appears that several notes are fragmented by the system (see Fig. 4.1), hence the low precision. In

note-based settings, the coarser granularity diminishes the risk of fragmentation, which improves the results, and in particular the precision. Recall slightly decreases in the note-based setting compared to the time-based one. This is likely due to the fact that the size of the note-based timesteps does not allow representing non-metrical notes properly. An analysis of the dataset has determined that around 13% of the note onsets in the dataset, when quantised to a twentieth of a quarter note, do not fall on the metric grid used. The decrease in recall is thus quite low compared to the proportion of notes that could theoretically be missed.

When inspecting the outputs of the LSTM (see Fig. 4.1), it appears that repetitions of a musical segment do not result in repeated outputs. We could expect an LSTM to enforce some consistency in terms of repeated outputs, but this is not the case. This could be due to the fact that the LSTM's input is a posterioqram estimated in a framewise fashion from the audio signal, and thus that it might vary from repetition to repetition, leading to different outputs. This also suggests that the LSTM is not able to keep track of its past outputs long enough to generate consistent repetitions of a sequence of notes.

Surprisingly, the HMM model yields quite poor results, lower than the baseline. It has the highest precision of all systems, but its recall is particularly poor. This could be due to the fact that the  $\lambda$  parameter of the HMM was optimised on the notewise F-Measure. Upon inspection of the outputs for various  $\lambda$ , it appears that when lowering  $\lambda$ , the HMM has a tendency to merge consecutive notes. To prevent that effect, a high  $\lambda$  had to be chosen, which explains the high precision and low recall.

### 4.2.5 Discussion

In this section, we have presented an LSTM-based system to transduct time-pitch posterioqrams into piano rolls, as a post processing step for AMT systems. We studied the influence of time-based (fixed length of 10ms) and note-based (musical length of a sixteenth note) timesteps, evaluated on an updated version of the MAPS dataset [Emiya et al., 2010], using metrical ground truth (see Appendix A). We compared our approach to a baseline model and an HMM-based model. Our approach outperformed both when evaluated on a frame basis, but was outperformed by the baseline approach when evaluated on a note basis. However, using note-based timesteps improved the results over time-based timesteps, which is an encouraging result towards using more musically-motivated system designs.

The improvement brought by the note-based timesteps is two-fold. It allows the system to better take into account dependencies between successive notes,

and it quantises the transcription. To determine the relative importance of both effects, we perform another experiment. We compare the results of the note-based setting with a new setting, where the computations are made using the time-based LSTM, and the binary outputs are quantised using a 16th note grid as a post-processing step (majority voting over the binary time frames). We refer to this second setting as *time quantised*. Results are compared in Table 4.2. The results are equivalent in terms of F-Measure for both settings for frame metrics, and even slightly better for the second setting with the note metrics. This suggests that the only improvement brought by the note-based timesteps is the quantisation of the output; it actually does not help modelling temporal dependencies, or at least, not in the current experiment. It might also indicate that the transduction model mostly relies on instantaneous dependencies rather than on the temporal evolution of the input. Using a more sophisticated architecture and data augmentation techniques might help the network make sense out of temporal dependencies and improve the results over a simple post-quantisation of the output.

| Setting                | $F_f$       | $P_f$       | $R_f$ | $F_{n,On}$  | $P_{n,On}$  | $R_{n,On}$ |
|------------------------|-------------|-------------|-------|-------------|-------------|------------|
| Note-based setting     | 67.1        | 65.9        | 71.0  | 62.2        | 59.6        | 67.0       |
| Time quantised setting | <b>67.2</b> | <b>68.9</b> | 67.4  | <b>62.8</b> | <b>63.4</b> | 64.2       |

Table 4.2: Comparison of transduction results using note-based processing, and time-based processing with post-quantisation of the outputs.

Given that the same piano models are present in the training and testing datasets, it is also possible that the network learns how to correct the errors the acoustic model makes on a specific piano and does not generalise to other piano models. This question will be investigated in Section 4.3.

Finally, a limitation of the note-based timesteps is their inability to represent notes with durations that are not an integer multiple of a sixteenth note. To take into account tuplets, we could use as timestep the greatest common divisor between all the note values we wish to represent. For instance, using a timestep of a 24th note would allow us to represent triplets as well as sixteenth notes. The more note values we want to take into account, the smaller the timestep, which could lead to the same problem as time-based timesteps: mostly self transitions, and no learning of temporal dependencies (see Section 3.3). Moreover, in our representation, we do not differentiate between note onsets and continuations. As a consequence, repeated notes are represented the same way as held notes. We assume that by using two different symbols, we could prevent over-fragmentation of notes in the output.

## 4.3 Comparing various architectures

### 4.3.1 Motivation and Objectives

From the previous results, we conclude that using 16th note timesteps improves performance, but mostly because of duration quantisation. Using 16th note timesteps in the transduction process does not seem to help the network learn about temporal dependencies. We thus aim here to investigate various design choices to improve the performance of our model.

In this section, we perform a comparative study of various neural-network-based methods for polyphonic sequence transduction, focusing on classical piano music. We systematically compare the influence of certain design choices, namely the network architecture, the type of outputs and training loss. In terms of network architecture, we investigate a convolutional neural network (CNN) architecture, and compare its performance to the previous LSTM model. For the type of outputs, we investigate binary neurons (see Section 2.8.5), and assess whether they can bring improvements in this context as suggested for symbolic music generation [Dong and Yang, 2018]. We compare them to the approach described in Section 5 consisting in training a network with real-valued outputs, and thresholding these values as a post-processing step. In addition, we propose a method to directly use the F-Measure as a training objective by using binary neurons instead of the more standard cross-entropy loss used with real-valued outputs in Section 5.

These design choices are evaluated using inputs from two different multi-pitch detection systems. We choose to deviate from the acoustic model used in the preliminary study in Section 4.2, and use more recent, arguably better acoustic models: a piano-specific acoustic model [Kelz et al., 2016] and a multi-instrument acoustic model [Bittner et al., 2017]. In particular, these models are trained and tested in more challenging and arguably more realistic conditions: the splits used have no overlap in terms of recording conditions, nor music pieces.

Although the conclusion of Section 4.2 is that 16th note timesteps do not clearly help in modelling temporal dependencies, they also do not deteriorate results. Besides, they reduce the input dimensionality compared to regular, smaller timesteps, and thus the required training and test time (between twice and four times as fast when comparing 40ms and 16th note steps, depending on the model and training loss). To allow a systematic comparison of all these configurations, all experiments in this section are thus conducted using 16th note timesteps.

The remainder of this section is organised as follows. In Section 4.3.2, we detail the architecture used for our various sub-models, and the training objec-

tives are described in Section 4.3.3. We present our evaluation metrics in Section 4.3.4, and describe our experiments and their results in Section 4.3.5. In Section 4.3.6, we investigate the influence of the timestep on the best-performing architecture. Finally, we discuss the limitations of our proposed methods in Section 4.3.7.

### 4.3.2 Models

#### Acoustic models

We use our transduction models with two different kinds of acoustic model outputs. The first model, described by Kelz et al. [2016], is a convolutional neural network (CNN) operating on spectrograms with logarithmically-spaced frequency bins and log-magnitude. It was designed specifically for piano transcription, and was trained on the MAPS dataset. At each timestep, it outputs predictions for  $N_p = 88$  pitches as probabilities between 0 and 1. We call this model *Kelz* (see also Section 2.3.2).

The second model, described by Bittner et al. [2017], also uses CNNs, with a custom input representation that stacks harmonically-shifted Constant-Q transform representations of the signal. It was designed as a general multi-pitch detection system, and was trained on a multi-instrument dataset that includes piano. Besides, it has a frequency resolution of 20 cents, and has a smaller frequency span than a piano keyboard. In order to adapt it for piano transcription, we average the frequency bins that correspond to a given MIDI pitch. More precisely, given  $p$  a MIDI pitch, and  $(c_i)$  the series of frequency bins used in [Bittner et al., 2017], we define  $C_p$  the subset of  $(c_i)$  such that  $c \in C_p \iff p - c < 50$  cents. Given  $P_B$  the output posterioqram from [Bittner et al., 2017], we then define:

$$P[p, t] = \frac{1}{|C_p|} \sum_{c \in C_p} P_B[c, t] \quad (4.4)$$

In this case,  $N_p = 73$ . It has to be noted that this averaging leads to overall lower activations. We call this model *Bittner*.

In both cases, outputs are then downsampled to a 16th note timestep. To do so, we use the same method as in Section 4.2, also using ground-truth 16th note annotations.

One major difference between the two acoustic models is that *Kelz* is trained on the MAPS dataset, while *Bittner* is not. In practice, *Kelz* overfits the training set, performing nearly perfectly. On the other hand, *Bittner*'s performance over all subsets is very stable. This means that for our post-processing model, training and testing conditions are much more similar for *Bittner* than for *Kelz*.

### Transduction model architectures

Our transduction models take the output of an acoustic model as input and are trained independently from the acoustic models. We compare two different kinds of transduction models. The first model is an LSTM network with the same architecture as described in Section 4.2. It has  $N_p$  inputs per timestep, one hidden layer with 100 hidden nodes, followed by a fully connected layer with  $N_p$  outputs.

The second model is a newly-designed CNN that uses a series of convolutions with filter sizes (height  $\times$  width):

1.  $5 \times 5$ , a classic CNN filter shape
2.  $5 \times 1$  with dilation of  $12 \times 1$ , which focuses specifically on octave errors
3.  $5 \times 15$ , which focuses on temporal correlations [Pons et al., 2016].

A  $1 \times 1$  convolution then processes the combined output from convolutions (2) and (3), whose output is combined with the output from convolution (1) using another  $1 \times 1$  convolution. While all above convolutions have 32 channels and use LeakyReLU activations, the final  $1 \times 1$  convolution has only one filter and does not use an activation. A schema of this architecture is given in Figure 4.2. Both models are closely matched in terms of the number of parameters at about 80,000.

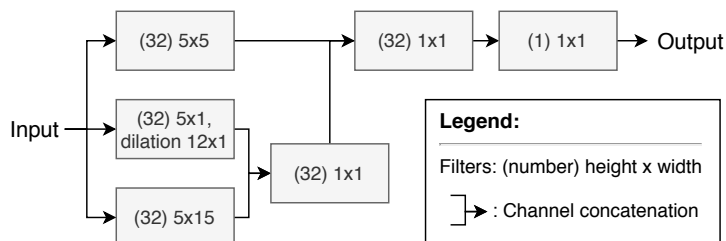


Figure 4.2: Architecture of the proposed CNN transduction model.

### Real-valued vs. Binary outputs

The output of the neural networks used in Section 4.2 is real-valued to enable training by gradient descent. To obtain a piano roll at test time, the network outputs are binarised with a threshold as a post-processing step. However, this can lead to overly-fragmented notes when the output values are close to the threshold, as discussed in Section 4.2.4. Binary neurons [Bengio et al., 2013], as described in Section 2.8.5, offer an alternative by integrating the binarisation of outputs into training while still allowing gradient back-propagation.

We thus compare two different strategies to convert the unnormalised, real-valued outputs obtained from the model. The first one is real-valued sigmoid outputs: the output of the network for each pitch is simply sent through a sigmoid. At test time, the outputs are binarised using a single threshold for all pitches, chosen to maximise  $F_f$  on the validation set. The second strategy employs deterministic binary neurons, as described by Dong and Yang [2018]. Here, the output of the network for each pitch is sent through a sigmoid, and then thresholded at 0.5. This thresholding operation, however, is not differentiable, which makes it impossible to use gradient descent as a training method. Therefore, we use the sigmoid-adjusted straight-through estimator [Chung et al., 2017], that was shown to provide good results [Dong and Yang, 2018]. It ignores the thresholding operation during back-propagation and instead multiplies the current gradient by the derivative of the sigmoid function.

### Baseline methods

We compare our system against two baseline post-processing methods. The first binarises all outputs using a threshold optimised as explained in Section 4.3.2 (as opposed to using a fixed threshold of 0.5 as done in [Kelz et al., 2016]). The second one is HMM smoothing [Poliner and Ellis, 2006], similar to Section 4.2. Contrary to 4.2, the acoustic model outputs have values in  $[0,1]$ , so they could be used directly be used as observation probabilities. However, results are poor when doing so, in particular with Bittner, due to its low output values. We thus choose to model the observations for ‘on’ and ‘off’ states as beta distributions, with probability density function defined on the interval  $[0, 1]$ :

$$Beta(\alpha, \beta) : prob(x|\alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)} \quad (4.5)$$

where  $B$  is the beta function

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt$$

The parameters of this distribution are fitted to the acoustic model outputs on the validation dataset, using data from all pitches and shared between them due to the rarity of extremely low and high pitches.

### 4.3.3 Training objectives

#### Cross-entropy loss

As a commonly used baseline objective, we use the binary cross-entropy (CE) loss averaged over all entries  $\hat{M}[p, t]$  in the predicted piano roll  $\hat{M}$  as loss for

the transducer model, as defined in Equation 4.2.

### F-Measure loss

In most cases, transduction and transcription models are trained with CE, but eventually evaluated with framewise or notewise F-Measure. To close this gap between training and testing, we propose maximising the framewise F-Measure directly during training. However, this requires a binary piano roll instead of real-valued network outputs. As a solution, we employ binary neurons introduced in Section 4.3.2. By replacing  $P_f$  and  $R_f$  in the formula for  $F_f$  by their complete expressions (see Section 2.6.2), simplifying the equation, we obtain:

$$F_f = \frac{2 \cdot TP}{2 \cdot TP + (FN + FP)} \quad (4.6)$$

where  $TP$ ,  $FP$  and  $FN$  are the numbers of true positives, false positives and false negatives respectively. We use the two following identities:

$$TP = \sum_{t=1}^T \sum_{p=1}^{N_p} M[t, p] \cdot \hat{M}[t, p] \quad FN + FP = \sum_{t=1}^T \sum_{p=1}^{N_p} |M[t, p] - \hat{M}[t, p]| \quad (4.7)$$

We finally obtain:

$$F_f = \frac{\sum_{t=1}^{N_t} \sum_{p=1}^{N_p} 2 \cdot M[t, p] \cdot \hat{M}[t, p]}{\sum_{t=1}^{N_t} \sum_{p=1}^{N_p} [2 \cdot M[t, p] \cdot \hat{M}[t, p] + |M[t, p] - \hat{M}[t, p]|]} \quad (4.8)$$

as our F-Measure loss, which only makes use of differentiable operators. The only exception is the absolute value function, whose gradient is undefined only at 0, which occurs in the above equation if  $M[t, p] = \hat{M}[t, p]$ . Since the model output  $\hat{M}[t, p]$  does not need to change in that case, we assign it a gradient of 0. In conjunction with binary outputs, we can thus use gradient descent to maximise  $F_f$ . It has to be noted that a similar, non-binary version of the F-Measure was proposed as training loss in [Pastor-Pellicer et al., 2013]. We only investigate the binary F-Measure here.

#### 4.3.4 Evaluation metrics

We evaluate the performance of our system using the framewise and onset-only notewise evaluation metrics described in Section 2.6.2. With framewise metrics, the metrics are computed directly on the 16th-note-timestep piano rolls, we do not convert them back to time-based timesteps, as the relative performance of the systems would not change. With notewise metrics, we compare the 16th note outputs with the 16th note targets. The threshold usually used is 50ms,

however, since we operate on 16th note timesteps, we require that onset times correspond exactly to the ground truth. The offsets are once again not taken into account as they are difficult to estimate properly with percussive sounds such as piano notes. Similarly to 4.2.3, since we do not estimate note onsets separately, both the output and ground truth lists of notes are obtained from piano rolls. We use the `mir_eval` implementation [Raffel et al., 2014] to find the maximal match between the two lists. These metrics are computed for each recording, and then averaged over sets of recordings.

### 4.3.5 Experiments

| Metric     | Thresh | HMM         | Cross-entropy |             | F-Measure |             |
|------------|--------|-------------|---------------|-------------|-----------|-------------|
|            |        |             | LSTM          | CNN         | LSTM      | CNN         |
| $F_f$      | 77.1   | 74.9        | 75.8          | 77.5        | 76.1      | <b>78.0</b> |
| $P_f$      | 77.7   | 67.8        | 77.8          | <b>79.6</b> | 78.6      | 79.2        |
| $R_f$      | 78.3   | <b>86.3</b> | 75.5          | 76.8        | 75.4      | 78.2        |
| $F_{n,On}$ | 71.4   | 73.0        | 69.6          | 73.4        | 69.4      | <b>73.8</b> |
| $P_{n,On}$ | 69.6   | <b>80.1</b> | 67.3          | 70.0        | 65.4      | 70.1        |
| $R_{n,On}$ | 74.7   | 68.4        | 73.4          | 78.6        | 75.7      | <b>79.4</b> |

Table 4.3: Results as percentages for median models with Kelz input. Bold corresponds to best values across models.

| Metric     | Thresh | HMM         | Cross-entropy |             | F-Measure |             |
|------------|--------|-------------|---------------|-------------|-----------|-------------|
|            |        |             | LSTM          | CNN         | LSTM      | CNN         |
| $F_f$      | 65.0   | 61.7        | 58.9          | 69.8        | 55.3      | <b>70.6</b> |
| $P_f$      | 65.3   | 53.2        | 56.9          | 71.8        | 48.1      | <b>72.4</b> |
| $R_f$      | 68.7   | <b>78.9</b> | 64.5          | 69.8        | 69.5      | 71.0        |
| $F_{n,On}$ | 58.7   | 48.4        | 53.5          | 62.0        | 47.0      | <b>63.5</b> |
| $P_{n,On}$ | 57.7   | <b>63.7</b> | 50.7          | 56.9        | 40.0      | 59.5        |
| $R_{n,On}$ | 61.0   | 40.1        | 58.2          | <b>70.2</b> | 59.0      | 69.9        |

Table 4.4: Results as percentages for median models with Bittner input. Bold corresponds to best values across models.

### Setup

As in Section 4.2, we use the MAPS dataset [Emiya et al., 2010] and the A-MAPS rhythm annotations (see Appendix A). For these experiments, to address the problem of overlap in recording conditions between train and test sets we mentioned in Section 4.2.5, we split our dataset into training, validation and test

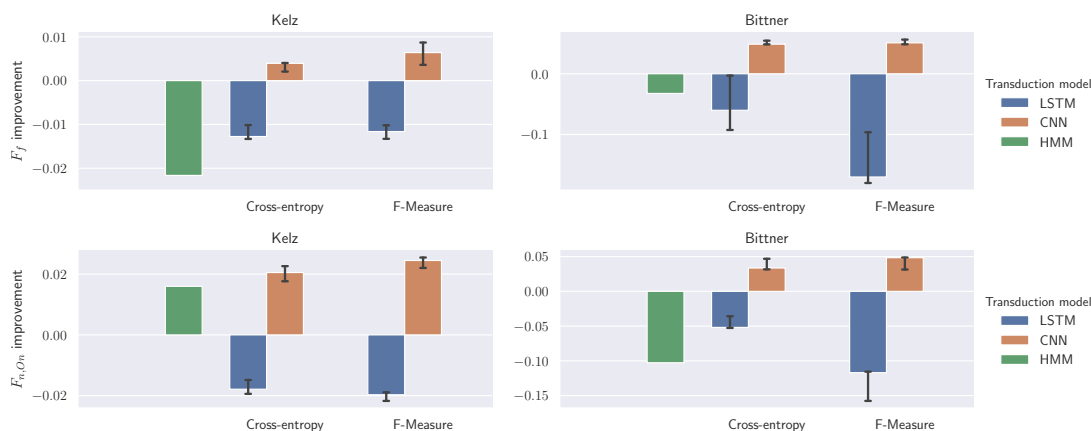


Figure 4.3: Model comparison across training objectives, transduction and acoustic models. Bars correspond to median improvement over simple thresholding, error bars correspond to maximum and minimum improvement across runs.

sets similarly to Kelz et al. [2016]. The test set features only acoustic piano and the training set only synthetic piano recordings, to obtain realistic performance estimates. We also ensure there is no overlap of pieces between the training and test set, and create a validation set by removing 10 random examples from the training set.

The networks are trained using the Adam optimiser [Kingma and Ba, 2015] on sequences of 64 timesteps. For cross entropy and F-Measure, we use a learning rate of  $10^{-2}$ , as advised in Section 4.2. Early stopping is used: after no improvement in validation  $F_f$  (see Section 4.3.4) for 30,000 iterations (checked every 1000 iterations), training is stopped and the best model is kept.

We evaluate every possible combination of acoustic and transduction model as well as training loss and output type (CE loss with continuous neurons or F-Measure loss with binary neurons). To account for variability in training, we run every experiment 3 times and report results of the model that reaches the median  $F_{n,On}$  performance. To assess differences in performance across different conditions, we first perform an independent-samples T-test for each piece, where each sample is the result of a specific run. Then, we test whether the resulting T-values are significantly different from 0. When comparing two baseline models, for which multiple runs are not available, we perform a paired-samples T-test. All median results are given in Table 4.3 for Kelz, and Table 4.4 for Bittner. We also plot the absolute improvement over simple thresholding for  $F_f$  and  $F_{n,On}$  in Figure 4.3. The error bars show the best and worst of the 3 runs.

## Results

First, we compare the transduction model architectures. The two transduction models (CNN and LSTM) clearly behave very differently. In every setting, the CNN model improves the results compared to simple thresholding. By contrast, the LSTM model performs worse than thresholding in every configuration. This contradicts the improvement in  $F_f$  reported in Section 4.2 achieved using an LSTM, although one could argue that the experimental setup is quite different regarding the LSTM inputs, the lengths of training sequences, and dataset splits. We tried using training sequences of 256 instead of 64 timesteps and also tried a bi-directional LSTM to check whether a lack of context explains our result, but results did not improve. Since we observe much better performance on the training set than on the test set (e.g. the LSTM with CE loss on the Bittner model achieves about 84% on training but only 58%  $F_f$  on the test set), we suspect that strong overfitting is the sole reason. This effect might be amplified as the training and test sets do not share any pieces or pianos, and synthetic pianos are used for training and real pianos for testing. Since the partitions used in Section 4.2 do share pianos, the improvement reported therein could be partly due to overfitting to the particular pianos featured in the training set. Our CNN appears to generalise better in this regard.

We now compare the two training losses used. Since our LSTM model does not generalise to the test set, we focus our comparison on the CNN model. It appears that with both acoustic models, using the F-Measure loss improves performance. With Kelz, it improves significantly for  $F_f$  ( $p = 0.02$ ), but not for  $F_{n,On}$  ( $p = 0.12$ ). With Bittner, it improves significantly for  $F_{n,On}$  ( $p = 0.045$ ), but not for  $F_f$  ( $p = 0.99$ ). This is an encouraging result, and shows that integrating the thresholding operation into the training process is at worst equivalent to, at best better than thresholding as a post-processing step.

We then compare the best-performing model on both inputs (i.e. CNN trained with F-Measure loss) with both baseline systems. With both acoustic models, our model improves significantly over both baselines, for both evaluation metrics, which shows the usefulness of our proposed post-processing method.

However, the model does not yield the same improvement on both acoustic models. The F-Measure-trained CNN improves over thresholding for Bittner by 5.6% in  $F_f$  and 4.8% in  $F_{n,On}$ , while for Kelz, it improves 0.9% in  $F_f$  and 2.4% in  $F_{n,On}$ . This difference in improvement can be explained by the fact that the Kelz acoustic model overfits a lot on the training set, reaching around 92%  $F_f$  on the training set alone. As a result, it is more difficult for transduction models to generalise with the Kelz model as input, as training and test data differ. The Bittner model, on the other hand, is much more stable across subsets, as it was

trained on a completely separate dataset, with 62%  $F_f$  on the training set.

Similarly to Section 4.2, the HMM post-processing performance is quite poor. On Kelz, it performs significantly worse than thresholding in terms of  $F_f$ , and not significantly better in  $F_{n,On}$  ( $p = 0.18$ ). On Bittner, it performs significantly worse than thresholding with both metrics. It seems to be too conservative, as shown by the high  $P_{n,On}$ : it outputs a note only when it is long enough and with high enough activation, a problem already noted in Section 4.2.

### 4.3.6 Comparison with time-based timesteps

We have found that the F-Measure-trained CNN is the best-working architecture on 16th-note steps, while the LSTM is not able to generalise to unseen recording conditions. It remains to be seen whether, with this better model, there is any difference in performance between using a 16th note processing step, or shorter, time-constant timesteps with an additional quantisation post-processing.

With the LSTM, the performance was similar in both cases, showing that it did not make use of temporal dependencies in the input. It might be that the network learned to only (or mostly) take into account the current frame in its hidden representation, and discards previous frames. The network could then be considered as a feed-forward network, made of two layers: the “write” gate and the output fully-connected layer. This would partly explain that it overfits the input acoustic conditions: feed-forward networks are known to be prone to over-fitting [Hinton et al., 2012a]. On the other hand, our CNN is designed to more explicitly take into account the temporal context by using long filter shapes in the temporal domain [Pons et al., 2016].

In order to test that assumption, we run one more experiment. We train a CNN with F-measure loss on inputs using a 40ms timesteps, for both acoustic models. For simplicity, we train only one transduction model for each acoustic model. We use the exact same setup as described in 4.3.5, except that we use a sequence length of 300 steps, so that the time duration of training sequences is of the same order.

We evaluate our models using 40ms steps, as described in 4.2.3. This allows us to also compare our results to the baseline in the absence of beat annotations, and to make sure that, despite the fact that note-based timesteps cannot represent all the durations found in the dataset, they are still useful. We report the results of 5 configurations:

**Baseline:** acoustic model outputs with time-based timesteps, thresholded with a threshold value tuned on the validation set;

**Baseline-quant:** Baseline outputs, quantised to a 16th note grid as a post-processing step;

**Time:** time-based transduction model, as is;

**Time-quant:** time-based transduction model, with outputs quantised to a 16th note grid as a post-processing step;

**Note:** note-based transduction model, with outputs upsampled back to time-based timesteps.

We exclude the HMM from this comparison, given the quite poor results obtained in previous experiments. For these 5 configurations, results are reported in Table 4.5 for Kelz, and Table 4.6 for Bittner.

First of all, our model improves significantly over the baseline: for both acoustic models, Time is significantly better than Baseline for both metrics, and both Time-quant and Note are significantly better than Baseline-quant. This shows that, overall, our proposed system is a useful and robust post-processing method.

When investigating the impact of quantising durations, we can see that, similarly to Section 5, quantisation of the outputs improves results for both acoustic models, in particular  $F_{n,On}$ : for both Kelz and Bittner, Baseline-quant is significantly better than Baseline in terms of  $F_{n,On}$  ( $p < 0.005$ ). Similarly,  $F_{n,On}$  is significantly better for Time-quant than for Time in both cases ( $p < 0.01$ ), although  $F_f$  is significantly worse. This confirms the fact that quantising durations helps performance, at least when using ground truth rhythm information.

Finally, we investigate how useful running computations with note-based timesteps is by comparing Note and Time-quant. In this case, results are mixed. For Kelz,  $F_{n,On}$  is significantly better with Note than Time-quant ( $p < 0.01$ ), showing that the model actually benefits from running computations on note-based timesteps. For Bittner, however, results are slightly worse with Note, although not significantly ( $p = 0.45$ ). In both cases,  $F_f$  is not significantly different between Note and Time-quant. We hypothesise that the fact that the improvement is higher with Kelz is related to the form of the training data. In the Kelz training set, the training data is nearly perfect, and very confident: when  $M[p, t] = 1$ ,  $P[p, t]$  is usually very close to 1, and usually very close to 0 when  $M[p, t] = 0$ . Overall, there are very few things to correct. The network probably learns more about note patterns than about how to correct erroneous detections by the acoustic model. Using note-based timesteps thus helps in this case, as note patterns can be represented over longer timescales than with time-based timesteps.

When evaluating the note-based and time-based versions of our proposed CNN with note-based timesteps (i.e. as described in Section 4.3.4), we observe significant improvement over time-based timesteps processing, both with  $F_f$

and  $F_{n,On}$ , and with both acoustic models ( $p < 0.05$  with a paired T-test). This shows that on note-based timesteps, our CNN benefits from processing inputs with musically-relevant timesteps, but this improvement is not necessarily preserved when comparing the outputs against the original time-based ground truth.

Overall, using note-based timesteps for computations either helps, or does not decrease significantly performance. This seems to indicate that overall, it is useful with this transduction model, while with the LSTM, it was not.

| Metric     | Baseline | Baseline-quant | Time        | Time-quant  | Note        |
|------------|----------|----------------|-------------|-------------|-------------|
| $F_f$      | 69.2     | 69.4           | <b>73.8</b> | 71.4        | 71.9        |
| $P_f$      | 69.6     | 72.4           | 74.7        | <b>75.8</b> | 70.6        |
| $R_f$      | 70.3     | 68.2           | 74.0        | 68.7        | <b>74.8</b> |
| $F_{n,On}$ | 56.7     | 64.1           | 66.4        | 70.2        | <b>71.9</b> |
| $P_{n,On}$ | 49.4     | 67.0           | 59.5        | <b>74.3</b> | 71.1        |
| $R_{n,On}$ | 70.5     | 63.5           | <b>78.4</b> | 68.4        | 74.1        |

Table 4.5: Comparison of time-based and note-based timesteps for the F-measure-trained CNN, all evaluated with time-based timesteps, with Kelz input. Bold corresponds to best values across configurations.

| Metric     | Baseline | Baseline-quant | Time        | Time-quant  | Note |
|------------|----------|----------------|-------------|-------------|------|
| $F_f$      | 59.3     | 59.2           | <b>67.0</b> | 65.7        | 65.4 |
| $P_f$      | 59.4     | 62.1           | 67.6        | <b>69.1</b> | 65.1 |
| $R_f$      | 63.5     | 60.3           | <b>68.5</b> | 64.4        | 68.2 |
| $F_{n,On}$ | 49.1     | 57.2           | 53.1        | <b>62.2</b> | 61.9 |
| $P_{n,On}$ | 42.6     | 59.3           | 44.7        | <b>64.3</b> | 60.5 |
| $R_{n,On}$ | 61.0     | 56.7           | <b>70.0</b> | 62.2        | 65.1 |

Table 4.6: Comparison of time-based and note-based timesteps for the F-measure-trained CNN, all evaluated with time-based timesteps, with Bittner input. Bold corresponds to best values across configurations.

### 4.3.7 Discussion

In this study, we showed that post-processing a posterigram with a convolutional network model can improve transcription performance compared to several baseline methods. Various other tasks include similar discretisation problems, such as polyphonic sound event detection [Cakır et al., 2017] or automatic drum transcription [Wu et al., 2018], and could probably benefit from this finding. In particular, we showed that using binary outputs for the CNN and training with  $F_f$  as objective can improve performance over using non-binary

outputs with CE loss and a thresholding post-processing step. We also showed that the results obtained in Section 4.2 actually do not generalise to the case where the recording conditions of the test and training datasets differ, highlighting the importance of carefully selecting dataset partitions in future work. Finally, we confirmed that using note-based timesteps with our CNN improves results compared to time-based timesteps beyond simple quantisation of the outputs, but only with one of the acoustic models (Kelz).

There are more things that we could have explored. More transduction architectures could be investigated, such as C-RNNs, or investigating other filter shapes for the CNN. Also, the improvement provided by the binary neurons is quite small, and not always significant; other gradient estimators could be explored, and could bring higher and more consistent improvements.

## 4.4 Conclusion and perspectives

In this section, we have explored polyphonic music sequence transduction. We have shown that, for an LSTM model, using 16th note timesteps improves performance compared to 10ms timesteps, but only because the outputs are quantised. We have further shown that this LSTM model is unable to generalise to unseen pianos or recording conditions. We have proposed a CNN model for transduction, and shown that it is able to successfully generalise to unseen recording conditions, for 2 distinct acoustic models, when evaluated on 16th note frames. Moreover, binary neurons are also a useful addition, allowing us to improve results in some cases, and never deteriorating them. We have also shown that, contrary to the LSTM, using note-based timesteps with our CNN architecture yields similar or better results than time-based timesteps, even when taking into account the fact that note-based steps quantise durations.

This difference in conclusions when comparing note- and time-based timesteps with the LSTM and CNN is interesting. Indeed, the LSTM, a model that is supposed to be good at modelling sequences, does not seem to benefit from temporal dependencies, while the CNN does. We have argued in Section 4.3.6 that this might be due to the fact that the CNN explicitly models temporal dependencies with its filter shapes, while the LSTM could learn to discard them. It would be interesting to validate whether that is the case, for instance by inspecting how the forget gate behaves in the LSTM, or looking into the filter weights for the CNN. Also, it would be interesting to further investigate why with Bittner inputs, the CNN does not benefit from using note-based timesteps. Explainability of neural networks is a tough and open problem [Mishra et al., 2017], outside the scope of this thesis.

Finally, in all our experiments, when using note-based timesteps, we consider

that the rhythmic ground truth is given. We made this choice to assess, as a proof-of-concept experiment, the improvement that note-based timesteps can bring in an ideal case. A real-life system would obviously have to rely on a beat tracking algorithm to use those timesteps. It remains to be seen whether we can still observe an improvement when using automatically estimated, potentially imperfect beat positions. Chapter 5 provides a partial answer in the context of MLM decoding, showing that estimated beat positions, although different from ground truth, do not yield significantly worse results, sometimes even better. While it would be worth investigating for polyphonic sequence transduction, we choose not to in this thesis, as improvements when using musically-relevant timesteps are relatively modest compared to the results presented in Section 5.4.

## Chapter 5

# Music Language Model Decoding

### 5.1 Introduction

Similarly to Chapter 4, the objective here is to obtain a symbolic representation from a non-binary posterigram. We investigate here another approach, more similar to how language models are used in ASR, that we call MLM decoding. Here, we use a polyphonic music prediction model, similar to those investigated in Chapter 3, to assess the likelihood of output sequences, with a workflow similar to that described in Section 2.3.4. In particular, this model is independent of the acoustic model used.

Some previous approaches have used that workflow [Sigtia et al., 2016, Wang et al., 2018], but have reported only modest improvements due to the use of an MLM (see also Section 2.5.3). We formulate hypotheses as to the reasons for that limited success, and propose solutions that we will investigate in the remainder of this chapter:

**Previous systems combine the MLM and acoustic model predictions statically.** Usually, the two priors are multiplied, in a static way [Sigtia et al., 2016, Wang et al., 2018]. However, depending on the cases, each system might perform better or worse. We propose to use an additional blending model, that essentially learns in which context should the MLM or the acoustic model be trusted more.

**Previous MLMs use inappropriate timesteps.** In [Sigtia et al., 2016], the MLM operates on 32ms timesteps, a duration much shorter than the typical duration of a note, and unrelated to the tempo of the piece being analysed. Motivated by the findings in Chapters 3 and 4, we propose to investigate the use

of musically-relevant timesteps instead, using ground-truth rhythm information as well as using automatically-estimated beat positions.

**Previous systems struggle to recover from early mistakes.** In previous systems, there is a discrepancy between MLM training and inference. Indeed, during training, the MLM learns to make predictions on perfect sequences. However, at test time, output sequences are imperfect, as they are obtained from the noisy acoustic model predictions. As a result, the model struggles to recover from early mistakes during inference. This issue was noted both in [Sigtia et al., 2016] and [Wang et al., 2018]. We propose to use scheduled sampling [Bengio et al., 2015] to make the MLM more robust at test time.

The remainder of this chapter is organised as follows. In Section 5.2, we describe our proposed system. In Section 5.3, we describe the experimental setup we use for our experiments. In Section 5.4, we discuss the results of our experiments. Finally, we discuss our results and propose some perspectives for improvement in Section 5.5.

This chapter is based on experiments presented at the International Society for Music Information Retrieval Conference (ISMIR 2019), reference (8) from Section 1.4, and extends them, in particular experimenting with automatically-detected beat positions.

## 5.2 Proposed System

Our AMT system flow is shown in Figure 5.1. It takes as input a spectrogram, with logarithmically-spaced frequency bins and log-magnitude with a timestep of 40ms. It consists of three main components: an acoustic model [Kelz et al., 2016], which predicts the presence of each pitch at a frame given the input spectrogram at that frame; an LSTM-based language model (see Section 5.2.2), which predicts the presence of each pitch at a frame given the previous frames; and a blending model (see Section 5.2.3), which combines the acoustic prior with the LSTM’s priors at each frame, and outputs a final combined probability distribution over pitches at each frame. The search process for finding the most probable output according to our system is detailed in Section 5.2.4.

The main contribution of this chapter lies in the post processing of the acoustic model’s outputs. These outputs, that are taken as inputs by our decoding method, are in the form of a matrix  $P \in \mathbb{R}^{N_p \times T}$ , where  $T$  is the length of the input in frames,  $N_p = 88$  (one row per key on a piano keyboard), and each element  $P[p, t]$  contains the probability of a pitch  $p$  being present at frame  $t$ . Our output is a binary matrix  $\hat{M} \in \{0, 1\}^{N_p \times T}$ , where  $\hat{M}[p, t]$  is 1 if pitch  $p$  is present at frame  $t$ , and 0 otherwise. In particular, this system does not output

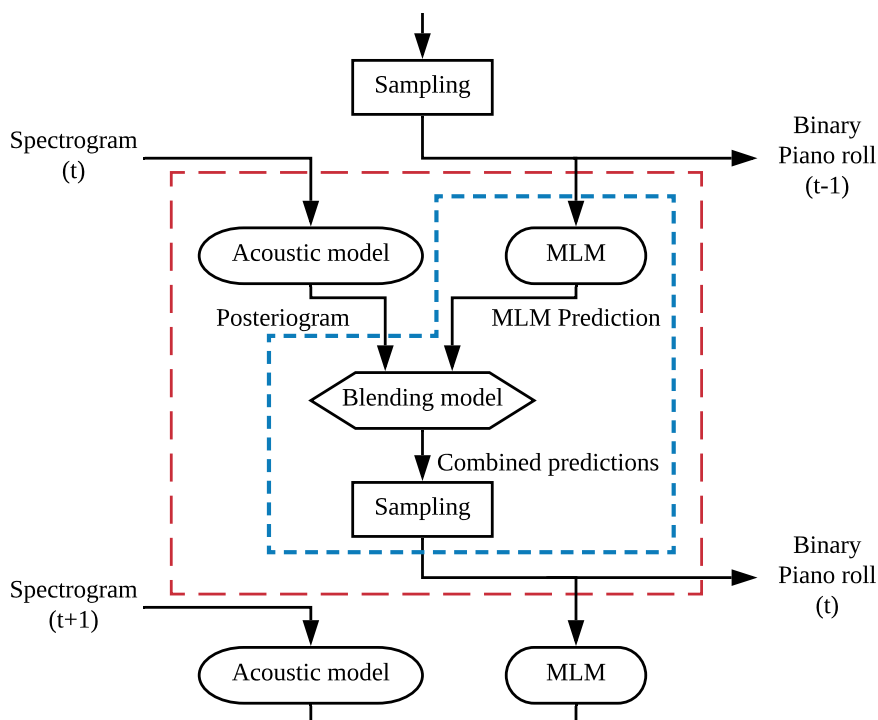


Figure 5.1: The proposed system for MLM decoding. The red box corresponds to one AMT processing step. The blue box corresponds to the decoding system, the main contribution of this chapter.

onsets explicitly: repeated and held notes are represented the same way.

### 5.2.1 Acoustic Model

We use as acoustic model the model described in [Kelz et al., 2016], without making any amendments to its architecture. This model, a CNN which outputs independent pitch probabilities at each time frame (see also Section 2.3.2), is a benchmark acoustic model for piano transcription.

### 5.2.2 Language Model

The language model has the same architecture as described in Chapter 3. It is a single-layer LSTM, with a hidden layer of size 256 and a fully-connected layer with sigmoid outputs of size  $N_p$ . It is trained to predict which pitches might be present in the next frame of a binary piano roll, given all the previous (binary) frames, using cross-entropy between the output of the network and the actual next frame as training loss, as described in Equation 3.5.

During a typical training procedure, the input sequences from which the network learns are taken directly from the ground truth. Such an MLM learns

to make predictions based only on perfect musical sequences. However, during inference, the MLM must make predictions based on potentially noisy sequences, as input frames are obtained from previous predictions and noisy acoustic multi-pitch detections. This discrepancy hinders performance of MLMs when used in an AMT setup, as noted in [Sigtia et al., 2016] and [Wang et al., 2018]. To solve this problem, we use scheduled sampling [Bengio et al., 2015]: during training, at each timestep, instead of always using the ground-truth frame, we randomly choose either the ground-truth frame (with probability  $p_{GT}$ ), or a frame sampled from the predictions made by the MLM at the previous timestep. Training starts with  $p_{GT} = 1$ , and  $p_{GT}$  is decreased as training progresses (see Section 5.3.6 for details), allowing the MLM to progressively become more robust to noisy inputs and recover from previous mistakes.

One limitation is that the noise the MLM adapts to is not the same the MLM sees at test time, since samples are drawn using the acoustic model’s outputs as well at test time. We could sample from a distribution that does the same during training, but we choose not to, because (1) Bengio et al. [2015] mention that even adding uniform noise helps performance, so exactly matching noise distributions is less important, and (2) this would require paired audio and MIDI data for MLM training (as acoustic model predictions must be made from audio), which is available in smaller quantities than MIDI data alone.

In total, 8 different MLMs are trained: for each of the 4 timesteps we investigate (see Section 5.3.2), we train one MLM with scheduled sampling, and one without. We use  $L[p, t]$  to denote the MLM’s output corresponding to pitch  $p$  at frame  $t$ .

### 5.2.3 Blending Model

The intuition behind the blending model is that the MLM and the acoustic model might each perform better or worse in certain situations, so combining their probabilities with a constant weight may achieve poor results. The blending model’s task is to learn the situations in which each model performs well, and output the combined prior for each pitch and timestep based on both the probabilities from the acoustic model and the MLM, as well as some surrounding context.

The blending model is a feed-forward neural network with  $l$  hidden layers<sup>1</sup> with 5 nodes each followed by an output layer with a single sigmoid. For each pitch  $p$  at time  $t$ , it takes as input:

1. The acoustic and language priors at that pitch and frame ( $P[p, t]$  and  $L[p, t]$ ),

---

<sup>1</sup> See Section 5.3.6 for details on the training of  $l$  and *hist*.

| Feature | Description | Equation  |
|---------|-------------|---|
| 1-2     | Uncertainty | $\sum_{p' < N_p} \begin{cases} (1 - P[p', t])^2 & P[p', t] > 0.5 \\ (P[p', t])^2 & P[p', t] \leq 0.5 \end{cases}$ |
| 3-4     | Entropy     | $\frac{1}{\log_2(N_p)} \sum_{p' < N_p \wedge P[p', t] \neq 0} -P[p', t] \log_2(P[p', t])$                         |
| 5-6     | Mean        | $\sum_{p' < N_p} \frac{P[p', t]}{N_p}$  |
| 7-8     | Flux        | $P[p, t] - P[p, t]$   |
| 9       | Pitch       | $\frac{p}{N_p}$   |

Table 5.1: Definition of the features used for the blending model. For equations written using  $P$ , the second feature is calculated identically with  $L$ .

- The sample history at that pitch for the previous *hist* frames<sup>1</sup> (*i.e.*  $\hat{M}[p, t']$  for  $\max(0, t - \text{hist}) \leq t' < t$ ),
- Nine additional hand-crafted features, described in Table 5.1. These features are included in order to give the blending model some information about the distribution of the acoustic and language priors without having to feed the entire distribution into the model, thus keeping its number of parameters low. We did not study their importance in the success of this method (see Section 5.5).

The resulting input vector is of length  $11 + \text{hist}$ . We use one single model for all pitches.

The sample history allows the model to learn if there are certain situations in which the LSTM performs particularly well or poorly. Features 1-2 model how peaked the output distribution from each model is by computing how far from complete uncertainty (*i.e.* predictions are equal to 0.5) the model is. Features 3-4 also model how uncertain each model is, but with different nonlinear properties. Features 5-6 model the expected polyphony, features 7-8 model how fast-changing each model's predictions are, and feature 9 allows the blending model to learn if either model performs better or worse for high or low pitches.

We create two versions of the blending model. First, a weight model (WM) which outputs a weight  $w_{p,t}$ , which is used to calculate a blended prior  $B[p, t]$  as a weighted sum:  $B[p, t] = w_{p,t}P[p, t] + (1 - w_{p,t})L[p, t]$ . Second, a prior model (PM) which outputs  $B[p, t]$  directly. The main difference between the two models is that WM can only ever result in a  $B[p, t]$  that lies somewhere between  $P[p, t]$  and  $L[p, t]$ , while PM can always output any  $B[p, t]$  between 0 and 1.

### 5.2.4 Search Process

Since our model’s search space has a branching factor of  $2^{N_p}$  at each frame, we cannot perform a global search. Therefore, we use Viterbi decoding [Viterbi, 1967] with beam search using a beam of size  $b$  and a branching factor of  $k$ . Specifically, at each frame, we save only the  $b$  most probable histories up to that point. Then, for each of those histories at frame  $t$ , using the blending model’s output distribution  $B[p, t]$ , we sample the  $k$  most probable samples using Algorithm 2 from [Boulanger-Lewandowski et al., 2013], which allows us to enumerate samples from a vector of independent probabilities in order of decreasing likelihood. Again, we save only the top  $b$  from the  $b * k$  resulting hypotheses, and keep iterating. The sample at frame  $t$  is denoted  $S_t$ , and is a set containing the pitches active at that frame. The probability of a sample  $S_t$ , given the blended priors  $B[p, t]$  is:

$$P(S_t) = \prod_{p' \in S_t} B[p', t] \prod_{p' < N_p \wedge p' \notin S_t} 1 - B[p', t] \quad (5.1)$$

Beam search has the drawback that the beam can easily become saturated with only slight variations of the most probable hypothesis. Therefore, similar to [Sigtia et al., 2016] and [Korzeniewski and Widmer, 2018], we use a hashed beam search. We consider any two hypotheses which are identical for the past  $h$  frames to be duplicates of each other for our purposes, and only save the most probable of them.

The final output  $\hat{M}$  of our system is constructed using the sample history of the most probable state in the beam, such that  $\hat{M}[p, t]$  is 1 if  $p \in S_t$  and 0 otherwise.

## 5.3 Experimental setup

### 5.3.1 Data

For our experiments, we use the MAPS dataset [Emiya et al., 2010], which contains MIDI-aligned recordings of various classical music pieces, some as played by an upright Disklavier, and some synthesized using high-quality piano samples (see also Section 2.7.2). We use the exact same test set as was used in [Kelz et al., 2016] (which was created in the same way as *Configuration II* from [Sigtia et al., 2016], with the additional constraint that only the Disklavier recordings were used). We create our training and validation sets slightly differently because the blending model requires a reasonably-sized validation set on which to train. From the remaining synthesized pieces, we choose 20 to become the

validation set (counting multiple synthesized recordings of a single piece as only one piece), and use the remaining pieces for training. This results in final split sizes of 60 pieces for test, 105 for training, and 32 for validation. The decrease in training set size compared to [Kelz et al., 2016] does not seem to affect the performance of the acoustic model. We train the acoustic model on the whole pieces, but our evaluation is performed on the first 30 seconds of each recording, as is usually done, e.g. in [Kelz et al., 2016].

To train our MLM, we use MIDI files taken from the Piano-midi.de<sup>2</sup> dataset. The version of the dataset we use here contains 324 pieces of classical piano music from various composers, with both quantised note durations and expressive timings. Every piece in MAPS can be found in the Piano-midi.de dataset, as these files were used to create MAPS. To avoid training the MLM with pieces later used for testing, we split the dataset using the same pieces as in the MAPS splits: all the pieces in the MAPS test set are used for testing (52 pieces), all the pieces in the MAPS validation set are used for validation (20 pieces), and all the remaining pieces are used for training (252 pieces).

### 5.3.2 Timesteps

We use 4 different timestep configurations in our experiments:

**40ms:** the resolution of [Kelz et al., 2016].

**16th-note:** the input is divided into 16th-note frames based on the metrical grid, using the metrical annotations from the A-MAPS dataset (see Appendix A).

**GT-beat:** the input is first divided in beats, using ground-truth beat annotations. Then, each beat is divided into 6 non-identical frames, using the following subdivisions:  $0, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}$ . Such subdivisions, which were also used *e.g.* in [Akama, 2019], allow us to represent both 16th notes and triplets, while avoiding using a representation with too many repeated timesteps: there are only 6 timesteps per beat, compared to 12 if we used a constant timestep.

**EST-beat:** similar to GT-beat, but with beat positions estimated from audio using an RNN-based automatic beat tracking system [Böck and Schedl, 2011] and a comb-filter-based post-processing method [Böck et al., 2015], using the `madmom` [Böck et al., 2016a] implementation.

<sup>2</sup><http://piano-midi.de/>

We investigate the GT-beat and EST-beat steps because using 16th-note steps requires having both beat and meter annotations. Indeed, a beat typically corresponds either to four 16th-notes when the meter is binary, or 6 with compound meters. This information is difficult to estimate automatically, we thus use timesteps defined only from beat positions.

To downsample the acoustic prior for 16th-note, GT-beat and EST-beat timesteps, we take the average of its original 40ms frames for the duration of each new frame, as done in Section 4.2. Before evaluation, we upsample our outputs back to 40ms timesteps, assigning each resulting 40ms frame the value of the corresponding beat-related frame.

### 5.3.3 Configurations

Besides the two versions of our blending model described in Section 5.2.3 (WM and PM), we use a baseline blending model: a constant weight (CW) model, which always calculates  $B[p, t]$  similarly to WM, but using the constant  $w_{p,t} = 0.8$  for all  $p$  and  $t$  (a value set in an ad-hoc fashion on the validation set). CW should indicate whether the adaptability of the blending model is important for performance. For each blending model, we train a version both with and without MLM scheduled sampling (using +S to denote its use) for each timestep.

There is a risk with PM that the blending model might choose to dismiss the MLM input completely. With WM, even if the MLM is not used to choose the weight, it will still have an influence on the resulting probabilities, unless the blending model’s output is exactly 1. To see whether our improvement comes from the MLM or simply the use of the blending model, we also train a blending model which is identical to configuration PM, except that any of its inputs which come from the MLM’s predictions are set to 0 at both train time and test time (this includes the MLM prediction itself as well as various features which use the MLM’s output). We call this configuration PM-A, and present a brief discussion of its results in Section 5.4.3.

### 5.3.4 Metrics

We report framewise precision, recall and F-measure ( $P_f$ ,  $R_f$  and  $F_f$ ), onset-only metrics ( $P_{n,On}$ ,  $R_{n,On}$  and  $F_{n,On}$ ) and onset-offset metrics ( $P_{n,OnOff}$ ,  $R_{n,OnOff}$  and  $F_{n,OnOff}$ ), as described in Section 2.6.2. These metrics are all averaged across all recordings in the test set. In order to test whether a configuration is significantly better than another, we use a paired-samples T-test. The framewise metrics are computed by comparing the output piano roll to the ground truth piano roll, using 40ms frames (after upsampling when using 16th-note, GT-beat and EST-beat timesteps). For notewise metrics, since our

model does not output onsets and offsets explicitly, we treat any output 1 not preceded by a 1 as an onset, and any 0 not preceded by a 0 as an offset, similarly to Chapter 4. We treat the ground truth the same after first converting it into a piano roll. Thus, here, our “notewise” metrics do not correspond with notes exactly, but rather as close as our output format can get, as explained in Section 4.2.3. We use `mir_eval` [Raffel et al., 2014] to perform all calculations. We also perform two post-processing steps for the notewise metrics, for all methods: (1) minimum duration pruning, where we remove any notes shorter than 50ms; and (2) gap filling, where we fill rests shorter than 50ms.

As argued in [Hawthorne et al., 2018], the most relevant metrics are the notewise metrics. Indeed, a poor transcription system could still score high in terms of  $F_f$  if its only errors correspond to short spurious notes and fragmentation of held notes. When discussing our results, we thus concentrate mainly on the notewise metrics. Furthermore, the onset-only metrics are the most commonly-used ones for the task, and onsets are much more perceptually important (and salient) than offsets [Daniel et al., 2008, Böck et al., 2016b]. Thus, our main evaluation concentrates on onset metrics, and we discuss the onset-offset metrics only in Section 5.4.5.

### 5.3.5 Baselines

We compare our models against that of [Kelz et al., 2016], retrained with our training and validation sets. We threshold its output at 0.5, setting all values  $\geq 0.5$  to 1 and all others to 0. We refer to this method as *Kelz*.

We also compare our model against a common HMM baseline [Poliner and Ellis, 2006] where each pitch is represented by a simple 2-state (on/off) HMM, run independently, as described in Section 2.3.3. Typically, as observed in Chapter 4, the HMM raises precision and lowers recall, removing short spurious notes from the output. In [Poliner and Ellis, 2006], one HMM is trained per pitch class. For transposition invariance, we instead train a single HMM, and use it for all pitches. In previous work, it has been standard to use maximum likelihood estimation (MLE) to set the transition probabilities (by counting transitions in some training set), and to treat the input probabilistic piano roll directly as the observation probabilities. We instead propose to learn the transition probabilities with Bayesian Optimization (BO) [Mockus et al., 1978] to maximize  $F_{n,On}$  on the validation set. There are only two probability values to search for (since  $P(\text{off}|S) = 1 - P(\text{on}|S)$ ). We use the validation set instead of the larger training set so that the HMM has noisier observations during training (for both MLE and BO), and we set the initial state probabilities to a uniform distribution.

The resulting HMM is one which is much more likely to change states: for

40ms timesteps,  $P(\text{on}|\text{off})$  is 0.004 with MLE and 0.493 with BO, and  $P(\text{off}|\text{on})$  is 0.167 with MLE and 0.196 with BO. Other timesteps see a similar change. Specifically, the probability for transitioning from “off” to “on” is much greater, likely because the observed data is much more accurate at note onsets, and thus the model can safely trust those values in most cases. The BO-trained HMM leads to a significant increase in both  $F_f$  and  $F_{n,On}$  for all timesteps compared to the MLE-trained HMM. A comparison between the MLE-trained and BO-trained HMMs is given in Table 5.2. In what follows, we only report results of the BO-trained HMM, and we refer to it simply as HMM.

| Method    | 40ms timesteps |             |             |             |             |             | 16th note timesteps |             |             |             |             |             |
|-----------|----------------|-------------|-------------|-------------|-------------|-------------|---------------------|-------------|-------------|-------------|-------------|-------------|
|           | $P_f$          | $R_f$       | $F_f$       | $P_{n,On}$  | $R_{n,On}$  | $F_{n,On}$  | $P_f$               | $R_f$       | $F_f$       | $P_{n,On}$  | $R_{n,On}$  | $F_{n,On}$  |
| HMM (MLE) | <b>82.0</b>    | 55.5        | 65.4        | <b>74.6</b> | 49.0        | 56.6        | <b>87.2</b>         | 39.1        | 51.9        | <b>82.9</b> | 25.9        | 36.4        |
| HMM (BO)  | 78.8           | <b>65.1</b> | <b>70.9</b> | 63.5        | <b>66.4</b> | <b>63.7</b> | 80.8                | <b>72.5</b> | <b>76.0</b> | 75.5        | <b>69.6</b> | <b>71.5</b> |

| Method    | GT-beat timesteps |             |             |             |             |             | EST-beat timesteps |             |             |             |             |             |
|-----------|-------------------|-------------|-------------|-------------|-------------|-------------|--------------------|-------------|-------------|-------------|-------------|-------------|
|           | $P_f$             | $R_f$       | $F_f$       | $P_{n,On}$  | $R_{n,On}$  | $F_{n,On}$  | $P_f$              | $R_f$       | $F_f$       | $P_{n,On}$  | $R_{n,On}$  | $F_{n,On}$  |
| HMM (MLE) | <b>85.4</b>       | 46.5        | 58.8        | <b>81.0</b> | 33.5        | 44.7        | 69.2               | 39.7        | 49.4        | 61.6        | 29.0        | 36.9        |
| HMM (BO)  | 84.4              | <b>65.8</b> | <b>73.4</b> | 74.0        | <b>64.8</b> | <b>68.0</b> | <b>82.7</b>        | <b>62.0</b> | <b>70.3</b> | <b>71.2</b> | <b>64.1</b> | <b>65.8</b> |

Table 5.2: Comparison of MLE-trained and BO-trained HMMs, with all timesteps, with the best values in bold.

### 5.3.6 Training

The MLM is trained using the Adam optimizer [Kingma and Ba, 2015] with a learning rate of 0.001. Piano rolls are cut into smaller sequences of 750 frames for 40ms timesteps, and 300 frames for the 16th-note, GT-beat and EST-beat timesteps. When using EST-beat steps for MLM training, the estimated beat positions are computed on audio versions of the MIDI files, rendered using a simple soundfont (TimGM6mb). We augment the data by transposing each sequence by a number of semitones randomly chosen between -5 and 7 at each epoch, so that each tonality is equally represented without shifting the note range too much. We use early stopping, such that if the cross-entropy evaluated on the validation dataset does not decrease for 200 epochs, training is stopped, and the best model so far is kept. For scheduled sampling, we decrease  $p_{CT}$  linearly from 1 to 0.7 over 500 epochs. Validation is done using a fixed value of  $p_{CT} = 0.7$ , and we use early stopping once the schedule is finished (after 500 epochs).

The blending model is trained on the validation set. Training data is generated by running our MLM on the first 30 seconds of each piece in the validation set with a fixed weight of 0.8 and a beam size of 10. We save a data point—containing the priors, a sample history of length  $hist$ , and features—for each

(frame, pitch, hypothesis) triple for which the acoustic prior differs from the language prior by at least  $\Delta_{min}$ . For each pitch  $p$  and frame  $t$ , we define targets as follows:

**WM:** if the ground-truth piano-roll value  $M[p, t]$  is 1, we use 1 if  $P[p, t] > L[p, t]$ , and 0 if  $P[p, t] < L[p, t]$ . If  $M[p, t] = 0$ , we use 1 if  $P[p, t] < L[p, t]$ , and 0 if  $P[p, t] > L[p, t]$ .

**PM:** we use as target value the ground-truth piano-roll value  $M[p, t]$  directly.

The blending model is then trained using binary cross-entropy loss, and training is stopped when there is no training loss decrease for 10 epochs. Bayesian Optimization for 200 iterations is used to search for the values of  $\Delta_{min}$  and  $hist$  (up to 10 for 16th-note, 12 for GT-beat and EST-beat timesteps, and up to 50 for 40ms timesteps), and how many hidden layers  $l$  to use (1–4 of size 5), using  $F_{n,On}$  on the validation set as objective.

The parameters for the beam search are set in an ad hoc fashion on the validation set. The beam size  $b$  and the branching factor  $k$  have small effects, where larger values lead to better results, but slower computation, and we use  $b = 50$  and  $k = 5$  for evaluation. The hash length  $h$  has an effect where smaller values force the model to perform a more global search, but with less ability to make decisions based on frames further in the past. We use a value of  $h = 12$  for evaluation.

## 5.4 Results

Full framewise and onset-only notewise results can be found in Table 5.3. First, we compare the best performing model for each timestep to the baselines in Section 5.4.1. Then, we investigate the impact of each component of our system: the timestep in Section 5.4.2, the blending model in Section 5.4.3, and scheduled sampling in Section 5.4.4. The onset-offset notewise metrics are presented in Section 5.4.5. We also present some qualitative observations on some selected outputs in Section 5.4.6.

### 5.4.1 Comparison with baseline

When comparing our models with the baselines in terms of  $F_f$ , baseline systems outperform our proposed method for all timesteps: HMM outperforms all other methods for 40ms, 16th-note and GT-beat timesteps ( $p < 10^{-4}$ ). For EST-beat, Kelz outperforms HMM slightly, but significantly ( $p = 0.01$ ), and all other methods by a larger margin ( $p < 10^{-4}$ ). This is understandable, as our systems were optimised for  $F_{n,On}$  rather than for  $F_f$ .

| Method | 40ms timesteps |             |             |             |             |             | 16th note timesteps |             |             |             |             |             |
|--------|----------------|-------------|-------------|-------------|-------------|-------------|---------------------|-------------|-------------|-------------|-------------|-------------|
|        | $P_f$          | $R_f$       | $F_f$       | $P_{n,On}$  | $R_{n,On}$  | $F_{n,On}$  | $P_f$               | $R_f$       | $F_f$       | $P_{n,On}$  | $R_{n,On}$  | $F_{n,On}$  |
| Kelz   | 79.5           | 62.2        | 69.3        | 58.4        | 65.4        | 60.2        | 85.0                | 64.8        | 73.0        | 71.1        | 67.0        | 67.9        |
| HMM    | 78.8           | <b>65.1</b> | <b>70.9</b> | 63.5        | <b>66.4</b> | <b>63.7</b> | 80.8                | <b>72.5</b> | <b>76.0</b> | 75.5        | 69.6        | 71.5        |
| CW     | 80.3           | 61.7        | 69.3        | 60.6        | 64.5        | 61.2        | 86.7                | 61.2        | 70.9        | 76.7        | 59.0        | 65.3        |
| CW+S   | 80.5           | 61.6        | 69.3        | 61.9        | 63.9        | 61.6        | 86.8                | 61.1        | 70.9        | 75.5        | 59.3        | 65.0        |
| WM     | 79.9           | 61.9        | 69.3        | 62.3        | 62.1        | 61.2        | 86.9                | 61.7        | 71.3        | 75.7        | 64.5        | 68.7        |
| WM+S   | 81.4           | 59.0        | 67.7        | <b>67.6</b> | 60.7        | 62.9        | <b>87.2</b>         | 60.0        | 70.3        | <b>79.7</b> | 61.6        | 68.2        |
| PM     | <b>86.0</b>    | 47.0        | 58.7        | 58.5        | 60.0        | 56.3        | 86.6                | 62.7        | 72.1        | 72.1        | 69.4        | 69.8        |
| PM+S   | 81.1           | 59.8        | 68.2        | 66.6        | 61.5        | 62.8        | 85.9                | 62.0        | 71.3        | 76.0        | <b>71.3</b> | <b>72.8</b> |

| Method | GT-beat timesteps |             |             |             |             |             | EST-beat timesteps |             |             |             |             |             |
|--------|-------------------|-------------|-------------|-------------|-------------|-------------|--------------------|-------------|-------------|-------------|-------------|-------------|
|        | $P_f$             | $R_f$       | $F_f$       | $P_{n,On}$  | $R_{n,On}$  | $F_{n,On}$  | $P_f$              | $R_f$       | $F_f$       | $P_{n,On}$  | $R_{n,On}$  | $F_{n,On}$  |
| Kelz   | 84.7              | 64.7        | 72.8        | 68.6        | <b>65.6</b> | 65.9        | 82.2               | <b>62.7</b> | <b>70.6</b> | 67.0        | 67.2        | 65.4        |
| HMM    | 84.4              | <b>65.8</b> | <b>73.4</b> | 74.0        | 64.8        | <b>68.0</b> | 82.7               | 62.0        | 70.3        | 71.2        | 64.1        | 65.8        |
| CW     | 85.8              | 62.5        | 71.7        | 73.7        | 60.6        | 65.3        | 83.5               | 60.4        | 69.5        | 73.0        | 61.8        | 65.2        |
| CW+S   | 85.8              | 62.4        | 71.6        | 73.3        | 59.7        | 64.5        | 83.6               | 60.3        | 69.4        | 72.5        | 61.7        | 64.9        |
| WM     | 85.6              | 61.0        | 70.0        | 75.7        | 60.8        | 65.6        | 83.6               | 60.0        | 69.2        | 75.7        | 64.2        | 68.1        |
| WM+S   | 85.8              | 62.0        | 71.2        | <b>76.4</b> | 60.9        | 66.5        | <b>83.8</b>        | 59.9        | 69.2        | <b>76.1</b> | 63.4        | 67.8        |
| PM     | <b>87.8</b>       | 59.0        | 69.3        | 69.5        | 64.8        | 65.6        | 83.6               | 59.8        | 68.8        | 72.3        | 68.0        | <b>68.7</b> |
| PM+S   | 78.1              | 64.9        | 66.3        | 65.6        | 62.8        | 62.5        | 81.7               | 60.9        | 68.9        | 68.9        | <b>69.9</b> | 68.1        |

Table 5.3: Results of all experiments, with all timesteps, with the best values in bold. CW uses our constant weight model, WM uses the weight model, and PM uses the prior model. +S denotes the use of scheduled sampling in training the MLM.

In terms of  $F_{n,On}$ , results are mixed. For 40ms timesteps, HMM significantly outperforms all our models ( $p < 10^{-5}$ ), except WM+S and PM+S ( $p = 0.17$  and  $0.16$  respectively). For GT-timesteps, HMM outperforms all other models ( $p < 0.01$ ). On the other hand, for 16th-note timesteps, PM+S gives better results than HMM ( $p = 0.003$ ), and for EST-best timesteps, all our blending model methods outperform HMM ( $p < 10^{-4}$ ).

We see that the EST-best PM model improves by 5% compared to the 40ms HMM. This is particularly encouraging, as both methods use the same amount of data, since the EST-beat methods do not use any ground-truth rhythm annotations.

## 5.4.2 Timestep

It is clear that using musically-relevant timesteps improves the results, for all tested timesteps: for all methods,  $F_{n,On}$  is higher with 16th-note, GT-beat and EST-beat timesteps than 40ms (except for PM+S with GT-beat timesteps), and  $F_f$  also improves most of the time. Similar results were seen in Section 4.3 (though not in Section 4.2, where improvements were mostly due to quantisation of the outputs). Here, the increase in performance of Kelz is due entirely to this quantisation. However, when performing the same quantisation procedure to the output of the 40ms-timestep WM+S using a 16th-note grid, we see a  $F_{n,On}$  of only 66.3%, significantly worse than PM+S with a 16th-note timestep

( $p < 10^{-5}$ ). Similarly, with EST-beat timesteps, quantising these outputs using the same steps as EST-beat, we obtain a  $F_{n,On}$  of 66.1%, significantly worse than PM with EST-beat steps ( $p < 10^{-4}$ ). However, with GT-beat steps, quantising the output of the 40ms WM+S model gives a  $F_{n,On}$  of 64.1%, which is not significantly worse than the WM+S results computed directly with that timestep. This shows that for this system, much of the improvement with 16th-note and EST-beat timesteps is due to its ability to learn more musical patterns at that scale, particularly when the full system has the ability to take advantage of that knowledge by dynamically weighting the influence of the MLM.

It is particularly interesting that all EST-beat models either perform significantly better or not significantly worse than their GT-beat counterparts in terms of  $F_{n,On}$ . This is not because the beat estimations are perfect: computing the beat F-measure of the estimations (using the usual tolerance of 70ms) is of only 67.6%. Some examples of detected beat positions are given in Figures 5.2 to 5.5. One could have expected that using imperfect beat annotations would yield worse performance than using ground truth annotations. Instead, these imperfections seem to help performance.

To understand why that might be the case, we investigate the beat tracking estimations, and compare them to the ground truth. It happens quite often that the beat-tracking system makes octave errors, compared to what is written in the ground truth, sometimes even jumping between octaves within a piece. While this might not correspond to the ground-truth, it is still relevant from a musical point of view. The only cases where the beat detection fails completely is when there is no clear beat, so rhythm information is probably not very useful anyway. And even then, the detected beat durations might be irregular, but beat positions tend to correspond to note onsets.

We further look into the outputs of Kelz with GT-beat and EST-beat timesteps, and inspect the cases where EST-steps most improve, and where they most decrease performance. We use this simple configuration to focus on the influence of the timestep in terms of duration quantisation, not taking into account any potential effect coming from differences in processing by the MLM or blending model. By looking at these cases, we can see that the main reasons for improvement are the following:

- The detected tempo is higher than the ground truth, which allows note values shorter than a timestep to be better represented (better temporal resolution).
- The detected tempo is lower than the ground truth (still high enough to keep a sufficient temporal resolution), so outputs are better smoothed.

The main reasons for deterioration are similar, but the other way around: either

the detected tempo is too high, so notes are more fragmented, or the detected tempo is too low, so some note values cannot be properly represented. Since EST-beat systems perform similarly or better than GT-beat, we assume that overall, the estimated beats tend to be more adapted to the density of notes present in the audio, regardless to how they are notated in the ground truth, which improves performance, in particular when using a post-processing method that helps smoothing out mistakes. We show in Figures 5.2 and 5.3 some examples of cases where estimated beats give better results than ground truth beats, and some cases where they are worse in Figures 5.4 and 5.5.

In the following sections, we thus concentrate on the results with EST-beat timesteps, as they reach good performance, and correspond to the most realistic use case. We also report results with 16th-note timesteps, as they reach the best performance overall and allow us to see what can be achieved in an ideal case.

### 5.4.3 Blending model

The adaptability of WM and PM to dynamically give importance to the MLM or the acoustic model clearly allows them to outperform CW, both with EST-beat and 16th-note steps. For EST-beat steps, all blending model configurations significantly outperform CW in terms of  $F_{n,On}$  ( $p < 10^{-3}$ ), but not in terms of  $F_f$ . The same can be said for 16th-note steps, they significantly outperform CW for  $F_{n,On}$  ( $p < 10^{-6}$ ), while for  $F_f$ , only PM outperforms CW.

Moreover, in both cases, the wider output range of PM leads to improvement over WM. For 16th-note steps, PM is significantly better than WM and WM+S for both  $F_{n,On}$  and  $F_f$ , while PM+S is significantly better than WM and WM+S for  $F_{n,On}$ , but not than WM for  $F_f$ . Results are less clear for EST-beat steps: the only significant difference is when comparing PM to WM+S in terms of  $F_{n,On}$ , but it still seems that prior models reach better notewise performance.

To see whether the improvement with PM comes from the MLM or simply from the use of the blending model, we evaluate the PM-A configuration. With 16th-note steps, this model achieves an  $F_{n,On}$  of 67.3% with a 16th-note timestep, significantly worse than PM+S ( $p < 10^{-9}$ ). With EST-beat steps, its  $F_{n,On}$  is 66.8%. The difference is smaller than with 16th-note steps, but it is still significantly worse than PM ( $p < 10^{-4}$ ). This shows that both the MLM and the blending model play an important part in our system’s performance. In the following sections, we concentrate on PM results.

### 5.4.4 Scheduled sampling

When comparing PM and PM+S, results are mixed. For 16th-note steps, we can see that PM+S performs significantly better than PM for  $F_{n,On}$  ( $p < 10^{-7}$ ),

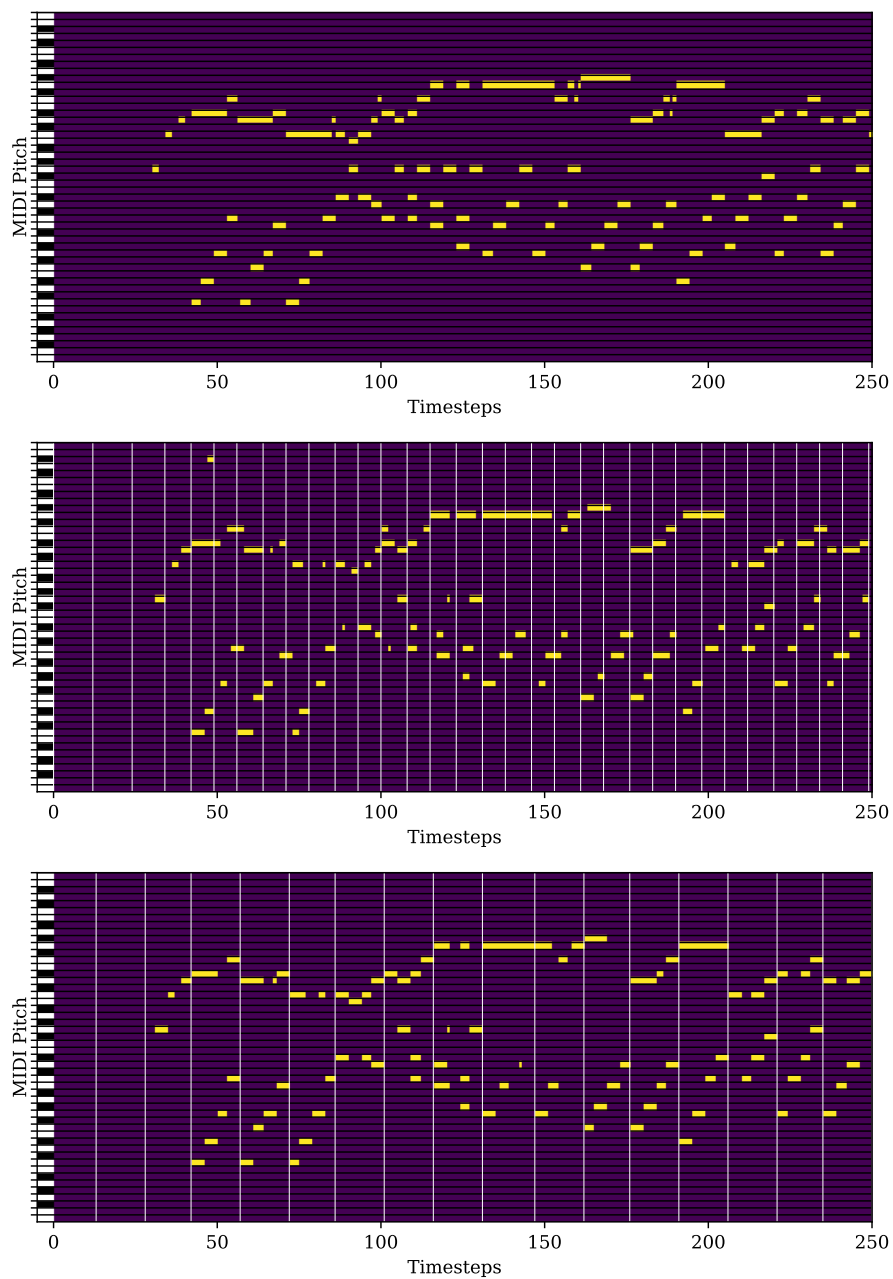


Figure 5.2: An example where EST-beat steps perform better than GT-beat steps. From top to bottom: ground-truth piano roll, GT-beat Kelz piano roll, EST-beat Kelz piano roll. Beat positions are marked as white vertical lines. EST-beat steps are longer, which helps smooth out spurious notes.

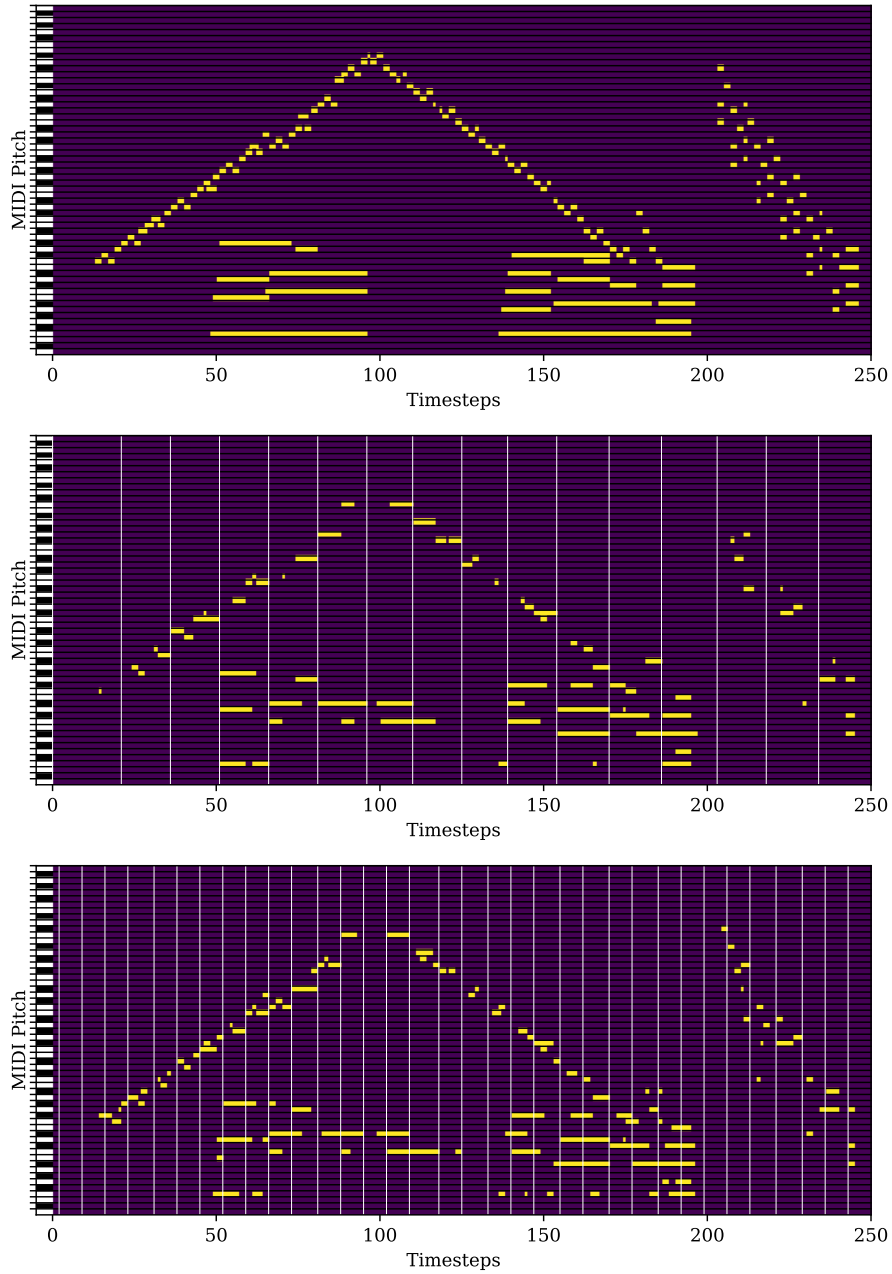


Figure 5.3: An example where EST-beat steps perform better than GT-beat steps. From top to bottom: ground-truth piano roll, GT-beat piano roll, EST-beat piano roll. Beat positions are marked as white vertical lines. EST-beat steps are shorter, which allows transcribing the short notes.

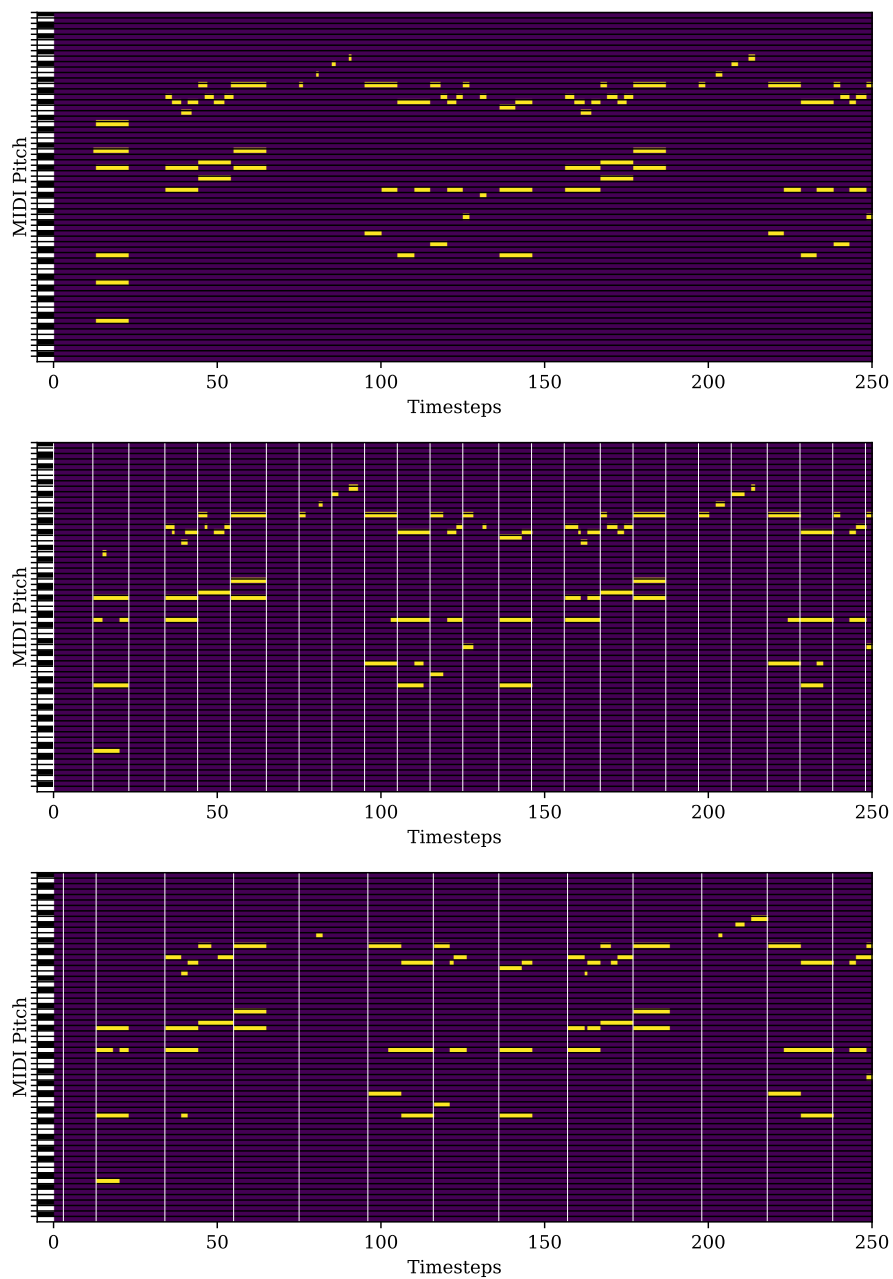


Figure 5.4: An example where EST-beat steps perform worse than GT-beat steps. From top to bottom: ground-truth piano roll, GT-beat Kelz piano roll, EST-beat Kelz piano roll. Beat positions are marked as white vertical lines. EST-beat steps are too long, so shorter notes are not properly represented.

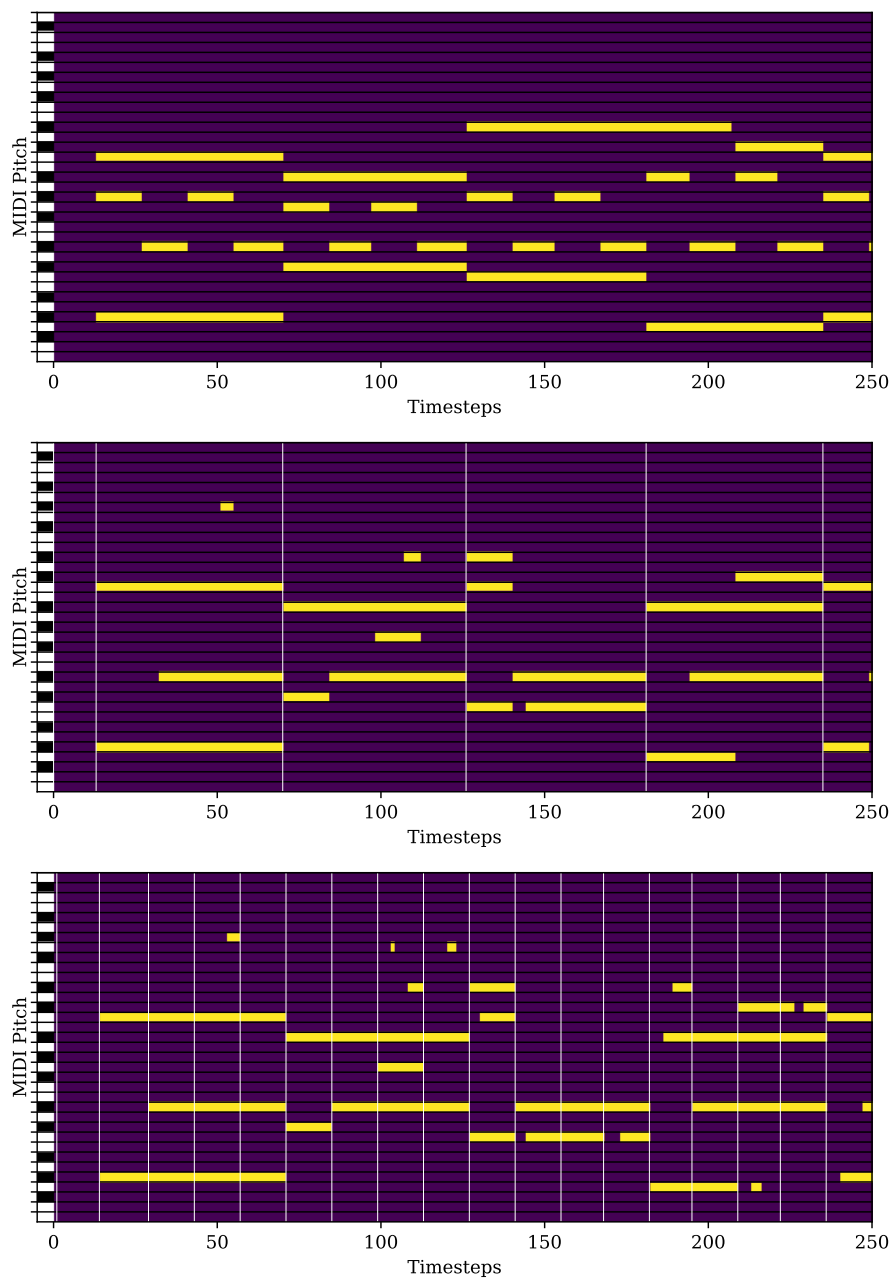


Figure 5.5: An example where EST-beat steps perform worse than GT-beat steps. From top to bottom: ground-truth piano roll, GT-beat Kelz piano roll, EST-beat Kelz piano roll. Beat positions are marked as white vertical lines. EST-beat steps are too short, which encourages spurious notes.

but PM performs better for  $F_f$  ( $p = 0.04$ ). On the other hand, for EST-beat steps, results are not significantly different, for both metrics ( $p = 0.22$  for  $F_f$  and  $p = 0.63$  for  $F_{n,On}$ ).

Looking at the results in a piecewise fashion gives us an insight on what scheduled sampling does. In Figures 5.6 and 5.7, we plot, for 16th-note and EST-beat steps respectively, the notewise  $F_{n,On}$  of Kelz (x-axis, a proxy for the noisiness of the input) against the increase in  $F_{n,On}$  for PM+S over PM (y-axis) for each piece in our test set. Here, it can be seen that PM+S outperforms PM by a greater margin in exactly those cases that we expect scheduled sampling to help: when the input is noisier. This correlation is significant for EST-beat ( $p = 0.02$ ), less so with 16th-note steps ( $p = 0.09$ ), and has high variance in both cases ( $R^2 < 0.1$  in both cases). The main difference is that with 16th note steps, scheduled sampling improves results most of the time, while with EST-beat steps, it improves results with noisy inputs, but deteriorates them when inputs are good, so we see no improvement on average.

Overall, we can conclude that scheduled sampling does indeed lead to improved performance with noisy inputs, but this does not necessarily result in an improvement when averaged over a whole dataset.

#### 5.4.5 Onset-offset Evaluation

Table 5.4 presents the OnOff-notewise results for the two baselines and the best performing model for each timestep. Our model significantly outperforms the baseline systems for 16th-note and EST-beat timesteps ( $p < 0.02$ ). For 40ms timesteps, our system outperforms Kelz ( $p < 10^{-5}$ ), but is not significantly different from the HMM ( $p = 0.54$ ). For GT-beat timesteps, our model is not significantly different from any of the baselines. This is a better result than with  $F_{n,On}$ , as with 40ms and GT-beat timesteps, our systems were significantly worse than baseline, which is encouraging. It seems that the MLM might have learned some rhythmic components of musical structure, allowing the system to correctly infer when notes end.

#### 5.4.6 Qualitative comparison

We look into some of the outputs of our best-performing model (PM+S with 16th-note timesteps), and compare them to baseline methods. In particular, we discuss one example where our model improves results, and one example where it degrades them. Figures and audio comparisons for these two examples can be found at the following address: <http://c4dm.eecs.qmul.ac.uk/ycart/ismir19.html>

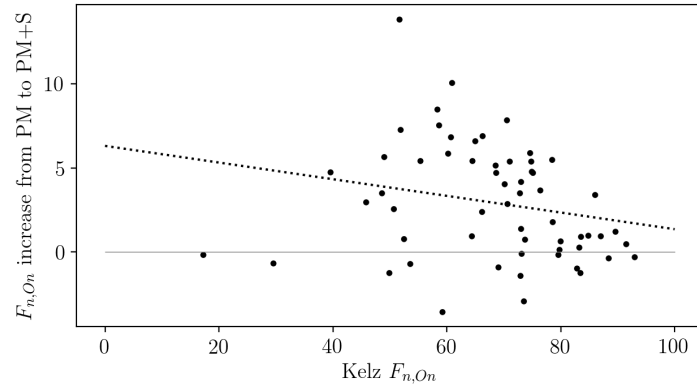


Figure 5.6: Increase of  $F_{n,On}$  of PM+S over PM plotted against  $F_{n,On}$  of Kelz for each piece, with 16th-note timesteps. Dotted line shows linear correlation ( $p = 0.09$ ,  $R^2 = 0.05$ ).

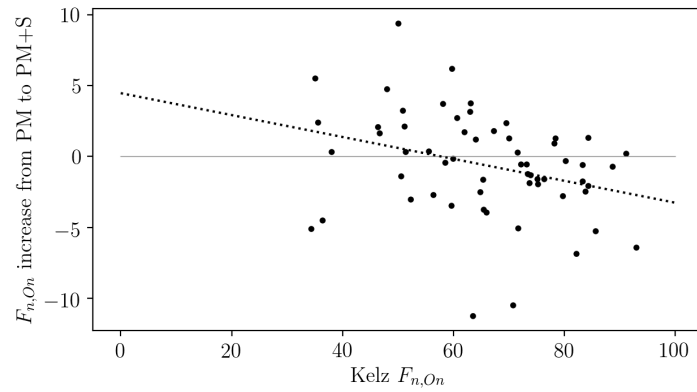


Figure 5.7: Increase of  $F_{n,On}$  of PM+S over PM plotted against  $F_{n,On}$  of Kelz for each piece, with EST-beat timesteps. Dotted line shows linear correlation ( $p = 0.02$ ,  $R^2 = 0.10$ ).

| Method | 40ms timesteps |               |               | 16th-note timesteps |               |               |
|--------|----------------|---------------|---------------|---------------------|---------------|---------------|
|        | $P_{n,OnOff}$  | $R_{n,OnOff}$ | $F_{n,OnOff}$ | $P_{n,OnOff}$       | $R_{n,OnOff}$ | $F_{n,OnOff}$ |
| Kelz   | 33.0           | 36.2          | 33.8          | 40.7                | 38.6          | 39.1          |
| HMM    | 37.2           | <b>38.7</b>   | <b>37.3</b>   | 42.4                | 39.7          | 40.6          |
| Best   | <b>39.7</b>    | 35.7          | 37.0          | <b>45.0</b>         | <b>42.6</b>   | <b>43.4</b>   |

| Method | GT-beat timesteps |               |               | EST-beat timesteps |               |               |
|--------|-------------------|---------------|---------------|--------------------|---------------|---------------|
|        | $P_{n,OnOff}$     | $R_{n,OnOff}$ | $F_{n,OnOff}$ | $P_{n,OnOff}$      | $R_{n,OnOff}$ | $F_{n,OnOff}$ |
| Kelz   | 40.0              | 38.0          | 38.4          | 38.8               | 38.3          | 37.7          |
| HMM    | 43.1              | <b>38.1</b>   | <b>39.9</b>   | 42.0               | 37.5          | 38.7          |
| Best   | <b>45.0</b>       | 37.0          | 39.8          | <b>42.8</b>        | <b>39.7</b>   | <b>40.4</b>   |

Table 5.4: Results using the onset-offset notewise metrics, using the best-performing configuration for each timestep (WM+S for 40ms, PM+S for 16th-note, WM+S for GT-beat and PM for EST-beat).

On the example where our system improves results, we can see that our system works better on repeated notes, as it manages to separate them out, while baseline methods tend to merge them together. Looking at the MLM predictions, it is clear that it helps in that regard: predictions for repeated notes and silence in-between are very confident, while they are much less clear on the posterigram.

On the other hand, on the example where it degrades the results, we can see that our systems tend to add more false positives, sometimes as octave errors, sometimes as over-fragmented notes. For this specific example, the MLM fails to make any confident predictions as to which notes might happen, instead outputting blurry predictions. This can be partly explained by the fact that this example contains a lot of chromaticism, which is quite uncommon in the dataset used for MLM training (containing mostly tonal music). As a result, the MLM struggles to identify patterns and infer which notes are likely to be present in this piece.

## 5.5 Conclusion and perspectives

In this chapter, we have presented a system for converting a posterigram output of an acoustic multi-pitch detection system into a binary piano roll. Our system consists of an LSTM-based MLM and a feed-forward neural blending model to combine the MLM outputs with those from the acoustic model. We have shown that our system performs significantly better in terms of  $F_{n,On}$  and  $F_{n,OnOff}$  than thresholding the posterigram, as well as post-processing it with a new strong baseline HMM. We have further shown that the use of a musically-

relevant timestep allows the MLM to learn musical structures better than with a 40ms timestep. Moreover, using timesteps based on automatically estimated beat positions still gives better results than 40ms, and, interestingly, better results than ground-truth beat positions. We have also shown that scheduled sampling helps the MLM perform better in the case of noisy inputs.

In order to investigate further how and when the blending model improves results, we could run an ablation study on the blending model, *i.e.* re-training it while removing some of its inputs. We have already done something similar with the PM-A configuration, by removing the MLM inputs; it would be interesting to do this more systematically, also removing the history, or some of the features, to see which of these inputs are most important to the success of our method.

The output representation of the system described above does not distinguish between repeated and held notes. As a result, repeated notes get merged together. In order to avoid that, it would be interesting to investigate other output representations that can explicitly model onsets of notes. One way to do that would be for instance to use a ternary piano roll representation, where for each time and pitch,  $M[p, t]$  can have 3 values, corresponding to onset, continuation, or silence. We would then need to adapt the MLM we use and the proposed blending model to make predictions based on this representation. Predicting onsets only based on MLM predictions would probably yield quite poor results, we would thus also need an acoustic model that is able to explicitly predict onset positions. Such systems are much less common, but some approaches exist, such as for instance [Kelz et al., 2019]. Making such modifications would allow us to obtain a complete, general audio-to-MIDI system for piano music.

## Chapter 6

# Evaluation of Automatic Music Transcription Systems

### 6.1 Introduction

As discussed in Section 2.6.3, current evaluation metrics are simplistic, and do not take into account musical aspects of the resulting transcriptions. In this chapter, we investigate to what extent the current evaluation metrics correlate to human perception of the quality of an automatic transcription. While previous approaches have used artificially-modified stimuli and focused on specific types of mistakes in isolation [Daniel et al., 2008], we choose to use real AMT system outputs, in order to maintain ecological validity, and study a wider range of features. We reframe the problem of AMT evaluation as a symbolic music similarity problem: we try to assess how similar to the target the output transcription sounds, rather than simply counting the number of incorrectly detected notes. We gather judgements of similarity by conducting a listening test, and use these answers to examine how human perception of AMT quality correlates with the evaluation metrics commonly used. We investigate what musical features are most important to raters, and use them to define a new metric, that correlates significantly better with human ratings than benchmark metrics.

Gathering similarity ratings in a meaningful way is not straightforward. In particular, inter-rater agreement is infamously low for music similarity tasks [Flexer and Grill, 2016]. One of the reasons, besides intrinsic disagreement between raters, is that it is a difficult and ill-defined task. Our main concern is thus to make the test as easy as possible. As argued by Allan et al. [2007],

the difficulty of rating the absolute similarity between two excerpts, be it on a continuous or Likert scale [Likert, 1932], leads to low inter-rater agreement, as different raters might use different scales, and these scales might evolve throughout the experiment. To avoid that problem, we choose to give raters a binary choice: given one reference excerpt, and two possible transcriptions of that excerpt, participants have to answer the question “Which transcription sounds most similar to the reference?”. Another reason that makes rating difficult is having to remember long excerpts for subsequent comparison. In order to make the task easier, such that participants can rely mostly on their working memory, we use short audio excerpts, which prevents us from drawing any conclusions on the similarity of longer excerpts. Since we are mostly interested in notes rather than timbre or sound quality, we can afford to run this study in more loosely controlled acoustic conditions. We thus ran this study online, in order to gather as much data as possible. A major concern is to make the test easily accessible; in particular, it is designed so participants can answer as many or as few questions as they want.

In accordance with the rest of this thesis, and as it is by far the most discussed sub-domain of AMT, we focus our study on Western classical piano music. The validity of the present study is thus limited to this instrument and style, and should not be generalised e.g. to singing voice, or jazz music.

The remainder of this chapter is organised as follows. We describe the design of the listening tests in Section 6.2. In Section 6.3, we analyse the results of the listening tests, and in particular the agreement between ratings and benchmark evaluation metrics. We then define a new metric based on musical features and analyse which features were most important to users in Section 6.4. Finally, we discuss our results in Section 6.5.

The results presented in this chapter were submitted to the journal *Transactions of the International Society for Music Information Retrieval (TISMIR)*, reference (11) from Section 1.4.

## 6.2 Study design

### 6.2.1 Stimulus design

We obtain automatic transcriptions using several benchmark AMT systems. Using the best systems available currently would have led to very similar transcription mistakes, as they are all based on the same underlying methods. Instead, we aim to use a diverse sample of commonly used AMT methodologies. We thus use the following systems, all described in Chapter 2:

**OAF:** The current state of the art based on neural networks [Hawthorne et al.,

2019], trained to jointly detect note onsets and pitches.

**CNN:** A simple framewise convolutional neural network [Kelz et al., 2016].

**NMF:** A piano-specific system, based on non-negative matrix factorisation [Cheng et al., 2016].

**STF:** A system based on handcrafted spectral and temporal features [Su and Yang, 2015].

CNN is a framewise system: at each timestep, it outputs a list of active pitches. This is equivalent to a piano roll, but requires post-processing to obtain a list of note events. To get note events, we consider any silence followed by a note as an onset (and vice versa for offsets), and apply gap-filling and short-note-pruning, both with a threshold of 80ms, corresponding to two processing frames in this system.

We use the pieces present in the MAPS dataset [Emiya et al., 2010] of MIDI-aligned piano recordings, as it remains the most common benchmark dataset for AMT. We use only the full music pieces in MAPS, with the two recording conditions that correspond to real piano recordings, namely ENSTDkCl (close-field recordings) and ENSTDkAm (ambient recordings), the two most commonly-used evaluation subsets. To preserve musical validity, we manually segment the pieces into musical phrases, so that each excerpt lasts between five and ten and roughly corresponds to a coherent, self-contained musical unit. We try as much as possible to keep an integer number of bars, using the A-MAPS bar and beat annotations (see Appendix A). When material within a piece is repeated without transposition, we only keep the first repetition. The start and end times of each segment are made available for future study (see Section 6.6). We keep duplicate pieces, recorded with two different recording conditions. Eventually, we obtain 1552 reference examples.

To be as consistent as possible in terms of timbre between the reference and the transcriptions, all example MIDI files were rendered using the Yamaha Disklavier Pro Grand Piano soundfont<sup>1</sup>. Some systems could not transcribe note velocities, so for uniformity, we used a default MIDI velocity of 100 for every note of the output transcriptions. We kept the original velocities when rendering references to be able to use them later on in the analysis, as most of the time they are available in the ground-truth files.

### 6.2.2 User data

Before answering questions, users read an information sheet and gave their consent for participating. We collected their age, gender, and whether they had

---

<sup>1</sup>Download link: <http://freepats.zenvoid.org/Piano/acoustic-grand-piano.html>

a hearing disability. They then had to answer questions from the Gold-MSI test [Müllensiefen et al., 2014] corresponding to the Perceptual Abilities and Musical Training subscales. Each user also had the option to give comments on the strategies they used and the aspects that were most important to them when choosing between transcriptions. All data was anonymised, and the procedure was approved by Queen Mary University of London’s ethics committee (reference QMREC2066).

### 6.2.3 Setup

The test was conducted online, as the main focus of this study was not sound quality, but rather the note content of the transcriptions. Participants were advised to do the test using good headphones, in a quiet environment. In what follows, we call a question a set  $\{\text{reference}, \text{transcription1}, \text{transcription2}\}$ , where *transcription1* and *transcription2* are two transcriptions of the reference, made by two different systems. There are six questions per reference, one for each unordered pair of AMT systems. For each question, participants were presented with one “reference” audio player, two “transcription” audio players, and were asked to answer the question “Which transcription sounds most similar to the reference?”, as a two-alternative forced choice (see Figure 6.1 for a screenshot of the interface). To strike a balance between comparison robustness and number of answered questions, each question was rated by four participants, taking care to balance the order (*transcription1*, *transcription2*) and (*transcription2*, *transcription1*) in which the two transcription players are presented in the interface. Participants were allowed to listen to each example as many times as they wanted; however, to encourage them to rely on perception rather than analytical thinking, we advised participants to listen to each example as few times as possible. A five-minute time limit was also included. For each question, participants could report if they knew the reference by ticking an additional “I know this piece” box.

While designing the test, it became apparent that in some instances, making a choice was very difficult, for instance when the two transcriptions were nearly identical, or different but equally poor. We did not want to include a third alternative (such as “I don’t know”, or “both transcriptions are equally similar to the target”), as this would have made it much more difficult to produce a meaningful analysis of the difficult cases. Instead, we added an extra question: “How difficult was it to answer the question?”, on a 5-point Likert scale [Likert, 1932] from “Very easy” to “Impossible”. Guidelines were given to answer this question in terms of number of listenings required for each file, difficulty of making a choice, and confidence in that choice.

Automatic Music Transcription Listening test [Home](#) [Profile](#) [Logout](#) [Any questions? Contact Adrien Ycart](#)

## Question 1

Reference:

▶ 0:00 / 0:09  I know this piece

---

Which transcription sounds most similar to the reference?

▶ 0:00 / 0:09       ▶ 0:00 / 0:09

How difficult was it to answer the question?

Very easy   Easy   Neutral   Difficult   Impossible

[Next question](#)

Difficulty scale:  
**Very easy:** 1 play, immediately obvious answer, very confident  
**Easy:** 1-2 plays, straightforward answer, confident  
**Neutral:** several plays, not easy to answer, moderately confident  
**Difficult:** many plays, hard decision, not very confident  
**Impossible:** many plays, arbitrary choice, no confidence

Figure 6.1: Screenshot of the listening test website.

Getting participants to spend 30 minutes or more on a listening test without compensation can be difficult. To allow more flexibility, we designed the test so that each participant could rate as many examples as they wanted. If we had randomly picked questions, given the large number of examples, it would have been very difficult to ensure that several people answered each question. Instead, questions were presented to participants using the following rules:

1. Each participant cannot hear a reference more than once.
2. Each question cannot be rated more than four times.
3. Each new question is chosen among remaining candidates using the following steps:
  - (a) Choose a reference among those that have already been seen by other participants, and have not been fully rated (*i.e.* at least one of the six questions using that reference has less than four answers).
  - (b) If no such reference is available, choose a random new question.
  - (c) Otherwise, choose a question using that reference that has already been answered by other participants.
  - (d) If no such question is available, choose a new question using the same reference.

When choosing a reference among those that have been seen by other participants (step 3.(a)), we skewed the random choice towards references that had

more answers, in order to maximise the number of fully-rated references (i.e. references for which all system pairs were rated by four participants). Thanks to this procedure, the size of the pool of examples adapted dynamically to the number of gathered answers.

### 6.2.4 Participants

In total, 186 people participated in our study (excluding the 40 people that registered but did not answer any questions): 126 males, 58 females and 2 non-binary, with a median age of 28. We did not make any selection on participants. Many of them were trained musicians, as the median Gold-MSI score is 5.06 on a scale from 1 to 7 (compared to 4.81 in the general population for the subscales considered [Müllensiefen et al., 2011]). The median number of answered questions was 20, with 22 participants answering 50 questions or more (up to several hundreds). Overall, we gathered 4501 answers, 1080 questions with four ratings, and 153 examples for which all pairs of systems were rated by four participants. Four participants reported a hearing disability, for a total of 53 answers. We decided to keep them anyway, as they amount for a small proportion of answers, and we are not interested in fine judgement about sound quality.

## 6.3 Results

In what follows, we analyse the results of the participants' ratings. We only keep questions for which four answers have been gathered. We keep all such questions, even when the corresponding example has not been rated for all pairs of systems. When comparing proportions (e.g. user preference, or agreement between raters and benchmark metrics), error bars are obtained by bootstrap analysis [Efron, 1992], resampling with the same dataset size 100 times. The standard deviation of bootstrapped results is displayed.

### 6.3.1 Benchmark system performance

First, we run the chosen systems on all the test files. We evaluate them using the benchmark metrics described in Section 2.6.2. Results are presented in Table 6.1. Notewise metrics were computed using the `mir_eval` Python library [Raffel et al., 2014].

As expected, OAF is by far the best of all, for all metrics. The second-best is NMF, which can also be explained by the fact that it was trained on that specific instrument model, while this piano model is new to the other systems. The CNN comes in third position, and STF comes last.

It has to be noted that these results vary quite a lot between the two subsets ENSTDkCl and ENSTDkAm: results are usually worse on ENSTDkAm, since it corresponds to ambient piano recordings, which are usually noisier. In particular, for NMF, which was trained on isolated notes played on ENSTDkCl,  $F_{n,On}$  drops from 76.1 to 55.6 on ENSTDkAm. For CNN and STF,  $F_{n,On}$  drops of around 5%. Interestingly, OAF works similarly on both subsets. This can be explained by the fact that it was trained on the MAESTRO dataset [Hawthorne et al., 2019], a dataset containing mostly concert piano recordings, in conditions arguably closer to ENSTDkAm.

It also appears that although the performance in  $F_f$  is within a relatively small range of values, there are much bigger differences in performance in terms of  $F_{n,On}$  and  $F_{n,OnOff}$ .

| System | $P_f$       | $R_f$       | $F_f$       | $P_{n,On}$  | $R_{n,On}$  | $F_{n,On}$  | $P_{n,OnOff}$ | $R_{n,OnOff}$ | $F_{n,OnOff}$ |
|--------|-------------|-------------|-------------|-------------|-------------|-------------|---------------|---------------|---------------|
| STF    | 67.2        | 60.0        | 62.7        | 49.8        | 32.0        | 38.3        | 16.5          | 11.3          | 13.2          |
| CNN    | 80.2        | 58.2        | 66.1        | 77.0        | 54.9        | 63.2        | 33.5          | 24.6          | 28.0          |
| NMF    | 71.3        | 63.3        | 66.4        | 79.6        | 57.0        | 65.7        | 35.7          | 26.4          | 30.0          |
| OAF    | <b>89.0</b> | <b>79.5</b> | <b>83.8</b> | <b>85.9</b> | <b>84.1</b> | <b>84.9</b> | <b>66.9</b>   | <b>65.5</b>   | <b>66.2</b>   |

Table 6.1: Benchmark evaluation metrics for all systems, evaluated on the MAPS subsets ENSTDkCl and ENSTDkAm, with best values in bold.

### 6.3.2 Perceptual ranking of systems

Using the ratings, we evaluate the systems from a perceptual point of view (pairwise results shown in Figure 6.2). The ratings are generally in accordance with the benchmark metrics: a system is preferred when its  $F_{n,On}$  is better (we focus on  $F_{n,On}$  as this metric correlates best with ratings, as discussed in Section 6.3.3). The relative ranking of the systems is also the same: OAF beats all other systems, NMF beats CNN and STF, and CNN beats STF. There seems to be a relation between the difference in benchmark metrics and the magnitude of the majority: for instance, OAF has a bigger majority when compared to STF than to NMF. But that is not strictly the case: although CNN is much better than STF in terms of  $F_{n,On}$  and  $F_{n,OnOff}$ , it is only preferred about 65% of the time.

### 6.3.3 Agreement between ratings and benchmark metrics

In this section, we assess the extent to which ratings agree with  $F_f$ ,  $F_{n,On}$  and  $F_{n,OnOff}$ . We also investigate what factors influence the agreement between raters and benchmark metrics. We define the agreement with a given metric as follows. For each given answer, we check whether the choice made by the

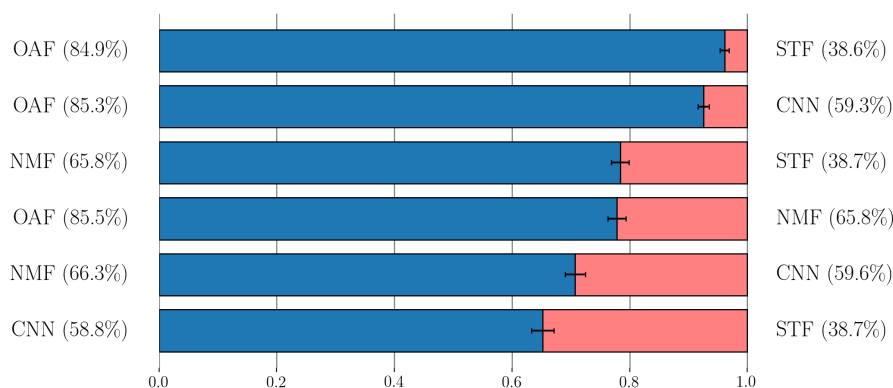


Figure 6.2: Vote proportion in pairwise comparisons of the systems. Blue bars represent the proportion of times the system on the left was chosen over the one on the right. For each pair, the percentage in brackets is the average  $F_{n,On}$  computed on the specific examples included in the comparison.

participant corresponds to the ordering of the two transcriptions according to this metric. If the participant chose the transcription for which the metric is highest, we consider that the participant and the metric agree. We then compute the proportion of ratings that agree with this metric. We do as such for  $F_f$ ,  $F_{n,On}$  and  $F_{n,OnOff}$ . For  $F_f$ , we investigate various frame sizes: 10, 50, 75, 100, and 150ms.

Results on the agreement between ratings and benchmark metrics are shown in Figures 6.3 and 6.4. In terms of frame size for  $F_f$ , there is no clear tendency. It does appear nonetheless that using a 100ms frame size improves slightly but significantly the agreement with ratings compared to a 10ms frame size ( $p < 10^{-3}$  with a Welch T-test). When examining the influence of the onset for  $F_{n,On}$ , we can see in Figure 6.3 that the agreement with ratings is highest for  $F_{n,On}$ , for onset thresholds between 75 and 150ms. For  $F_{n,OnOff}$ , we can see in Figure 6.4 that the agreement is highest for an onset threshold of 100ms and an offset tolerance of 50%, although it is still lower than  $F_{n,On}$  with onset threshold above 50ms. Agreement might be even higher for higher offset tolerance thresholds, as  $F_{n,OnOff}$  becomes more and more similar to  $F_{n,On}$  ( $F_{n,On}$  can be seen as  $F_{n,OnOff}$  with an infinite offset tolerance).

It has to be noted that the agreement varies greatly depending on the pair of systems being compared (see Figure 6.5). As in Section 6.3.2, when there is a large difference in performance between the two systems (e.g. OAF vs. STF), raters agree with  $F_{n,On}$ . Disagreement is highest when comparing the two systems ranked as worst (CNN vs. STF).

To investigate further which factors might influence agreement, we perform

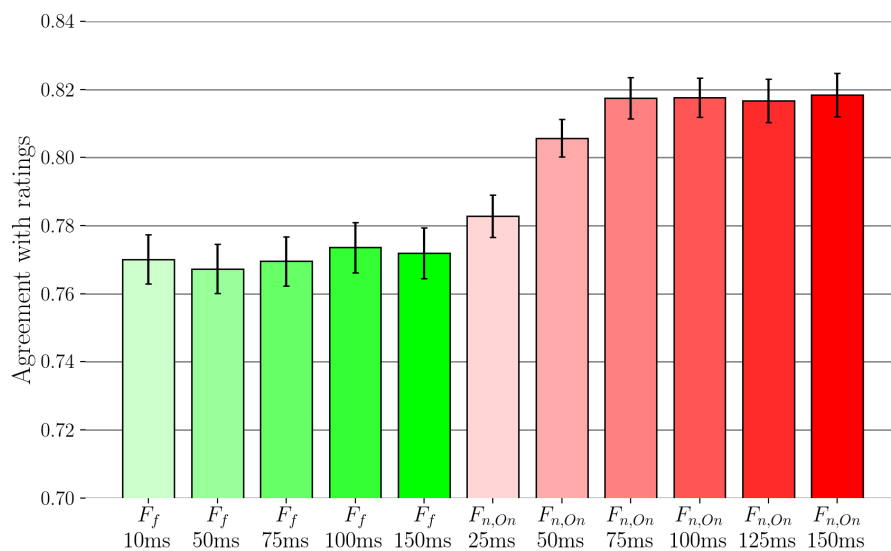


Figure 6.3: Percentage of agreement, across all examples, between raters and various evaluation metrics ( $F_f$  with various frame sizes, and  $F_{n,On}$  with various tolerance thresholds).

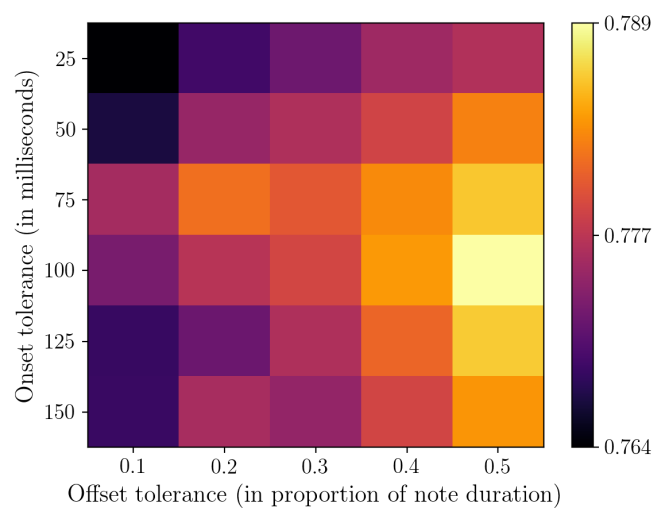


Figure 6.4: Percentage of agreement, across all examples, between raters and  $F_{n,OnOff}$ , with various onset and offset tolerance thresholds.

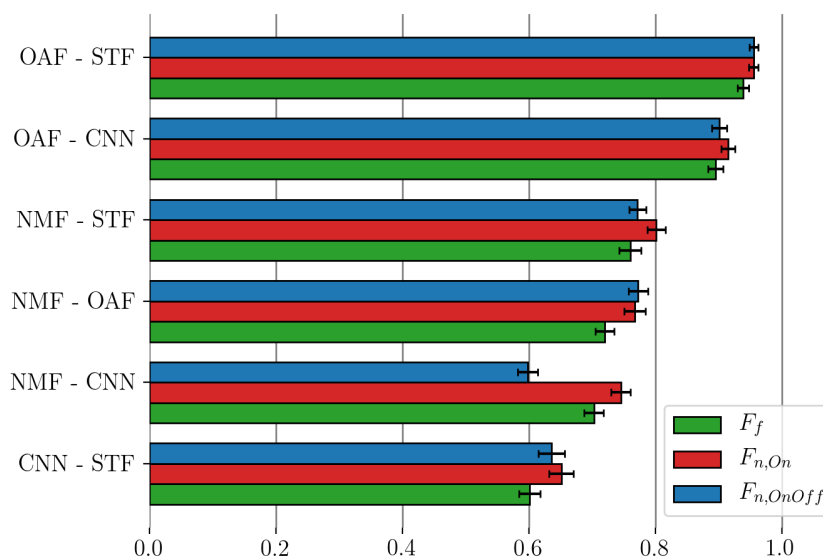


Figure 6.5: Percentage of agreement, for each pair of systems, between ratings and benchmark evaluation metrics.

a linear mixed effects analysis [Baayen et al., 2008], using as the dependent variable for each question whether the rater agreed with  $F_{n,On}$ . We use as fixed effects the best  $F_{n,On}$  of the pair ( $F_{best}$ ), the difference in  $F_{n,On}$  between the two transcriptions ( $\Delta F$ ), the Gold-MSI score of the rater (Gold-MSI), whether the piece was recognised (Known), and the reported difficulty (Difficulty). We use no random effects. The resulting coefficients and associated p-values are given in Table 6.2.

| Feature    | Coefficient | P-value |
|------------|-------------|---------|
| $\Delta F$ | 0.539       | <0.001  |
| $F_{best}$ | 0.330       | <0.001  |
| Gold-MSI   | -0.007      | 0.232   |
| Known      | 0.014       | 0.391   |
| Difficulty | -0.044      | <0.001  |

Table 6.2: Coefficients and p-values for the linear mixed effects model using agreement with  $F_{n,On}$  as dependent variable and features as fixed effects.

It appears that  $\Delta F$  and  $F_{best}$  have a strong and significant effect on agreement. When the difference in performance between the two systems is high, people tend to agree more with the F-measure, as the choice is clearer. However, for a given difference in  $F_{n,On}$ , when both systems produce outputs of poor quality, the agreement is lower.

When looking at other features, Difficulty is negatively correlated with agreement: when people report the choice as being more difficult, they tend to disagree more with the F-measure. To investigate this further, we compute the percentage of agreement between ratings and the F-measure for each reported difficulty level (Figure 6.6). For high levels of difficulty, agreement is very poor, close to chance (50% for a two-alternatives forced choice question), which is consistent with the guidelines given to raters for reporting difficulty. Still, even for low levels of reported difficulty, there is a fair amount of disagreement between ratings and  $F_{n,On}$  (10 to 20%), which shows that disagreement with  $F_{n,On}$  does not exclusively result from random choices in the difficult cases. Musical training (Gold-MSI) and familiarity (Known) have no significant effect on agreement with  $F_{n,On}$ .

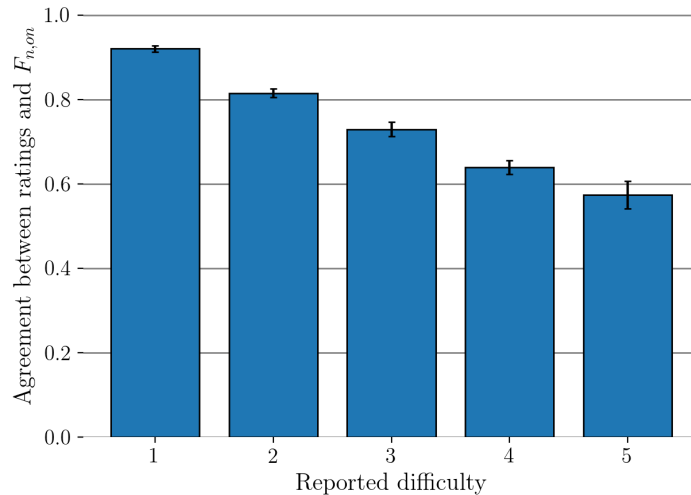


Figure 6.6: Agreement between ratings and  $F_{n,On}$  for each reported difficulty level.

### 6.3.4 Reported difficulty

In this section, we examine the reported level of difficulty for each answer, and investigate the factors that influenced it.

In Figure 6.7, we display the proportion of ratings for each difficulty level. When comparing this figure to the results in Table 6.1, it appears that, as a general trend, the higher the difference in  $F_{n,On}$ , the more confident raters are. Moreover, difficulty is highest when comparing the two worst performing systems according to benchmark metrics, which suggests that difficulty is higher when both transcriptions are poor.

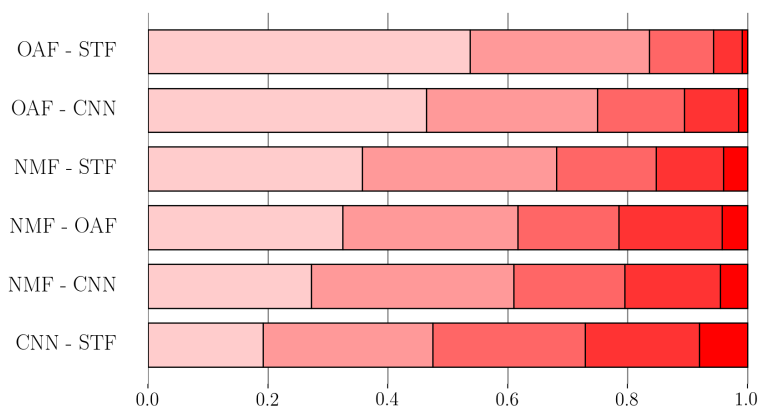


Figure 6.7: Proportion of difficulty ratings (lightest=1, darkest=5) for each pair of systems.

To get a better understanding of how the difficulty varies depending on various parameters, we perform another linear mixed effects analysis, using this time difficulty as dependent variable. We use as fixed effects the best  $F_{n,O_n}$  of the pair ( $F_{best}$ ), the difference in  $F_{n,O_n}$  between the two transcriptions ( $\Delta F$ ), the Gold-MSI score of the rater (Gold-MSI), whether the piece was recognised (Known), and whether the rater agreed with  $F_{n,O_n}$  (Agree). Again, we use no random effects. The resulting coefficients and associated p-values are given in Table 6.3.

All of these factors are significant predictors of reported difficulty. From this, we can draw the following conclusions. First, musicians found the task easier than non-musicians. This could be explained either in terms of better auditory skills, or because musicians tend to be more confident in their judgements. People also find it easier to make a choice when they know the reference. One user commented: “Songs that I knew already felt easier to judge as I could remember the original much better”, in other words they only had to listen to and remember two excerpts instead of three. This highlights a difficulty of investigating musical similarity perception due to effects of memory, as we mentioned in Section 6.1. It also appears that the more confident people are in their choices, the more they agree with the F-measure, which is coherent with the results presented in Section 6.3.3. Finally, when investigating the effect of  $\Delta F$  and  $F_{best}$ , we can see that the bigger the difference between the two systems, the easier the decision, and all the more so when both systems perform well.

To examine further the relation between  $\Delta F$  and Difficulty, we plot the distribution of  $\Delta F$  for each reported difficulty level in Figure 6.8. We can see that when  $\Delta F$  drops below 20%, the choice is often considered as difficult to

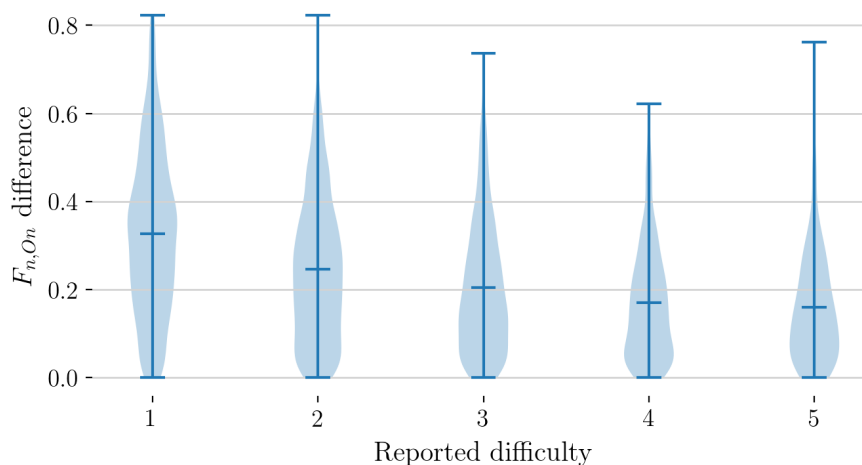


Figure 6.8: Distribution of difference in  $F_{n,O_n}$  between the two options for each reported difficulty level. The horizontal bar corresponds to the mean value.

make. Still, some cases with low  $\Delta F$  are also rated with difficulty 1 or 2, showing that even with low  $\Delta F$ , some differences in performance can be perceptually very clear.

| Feature    | Coefficient | P-value |
|------------|-------------|---------|
| $\Delta F$ | -1.564      | <0.001  |
| $F_{best}$ | -0.608      | <0.001  |
| Gold-MSI   | -0.227      | <0.001  |
| Known      | -0.153      | 0.002   |
| Agree      | -0.423      | <0.001  |

Table 6.3: Coefficients and p-values for the linear mixed effects model using difficulty as dependent variable and features as fixed effects.

### 6.3.5 Analysis of confident answers

When discussing the agreement between ratings and  $F_{n,O_n}$ , it is not straightforward to distinguish cases when participants chose randomly from cases where they actually disagreed with  $F_{n,O_n}$ , in particular where the two options have similar  $F_{n,O_n}$ , or when both options are poor. To avoid cases of random choice, we analyse the subset of answers that are confident (Difficulty = 1 or 2, which represents 2856 answers), and investigate whether different factors influence the agreement between ratings and  $F_{n,O_n}$  in this case.

We perform the same linear mixed effect analysis as in Section 6.3.3, on

that subset. The results are shown in Table 6.4 and are quite similar to the full analysis, except that now there is a significant negative correlation between Gold-MSI and agreement. For confident answers, it appears that musicians tend to disagree more with  $F_{n,O_n}$  than non-musicians. This could indicate that musicians focus more on certain high-level aspects of the music (e.g. melody, harmony, meter) that are not taken into account by  $F_{n,O_n}$ : even if it contains more mistakes, a transcription might be preferred by a musician as long as it gets these aspects right.

When investigating the effect of the difference in  $F_{n,O_n}$  on agreement, we see once again the same trend: the smaller the difference between the two transcriptions, the greater the disagreement, as shown in Figure 6.9. When the difference in  $F_{n,O_n}$  is above 50%, people always agree with  $F_{n,O_n}$ . However, below this threshold, agreement declines, especially when the difference is below 20%.

| Feature    | Coefficient | P-value |
|------------|-------------|---------|
| $\Delta F$ | 0.584       | <0.001  |
| $F_{best}$ | 349         | <0.001  |
| Gold-MSI   | -0.014      | 0.011   |
| Known      | 0.002       | 0.912   |
| Difficulty | -0.036      | <0.001  |

Table 6.4: Coefficients and p-values for the linear mixed effects model using agreement with  $F_{n,O_n}$  as dependent variable and features as fixed effects, on confident answers only.

### 6.3.6 Inter-rater agreement

We have seen that there is a fair amount of disagreement between the F-measure and ratings. To get an idea of how consistent the ratings are, we investigate the level of inter-rater agreement, and the factors that influence it.

We begin by computing Fleiss' Kappa coefficient [Fleiss, 1971], that represents inter-rater agreement for an arbitrary number of raters. When computed over the whole dataset, we obtain a Kappa coefficient of 0.59, which can be interpreted as borderline between moderate and substantial agreement. When computing the same coefficient on the confident answers only (keeping only questions for which four confident answers were given, 315 questions in total), we obtain a Kappa coefficient of 0.90, which can be interpreted as near-perfect agreement. This is a very conservative estimate, as we keep only the questions that were unanimously considered as easy to answer.

Moreover, inter-rater agreement is high because most of the time, raters

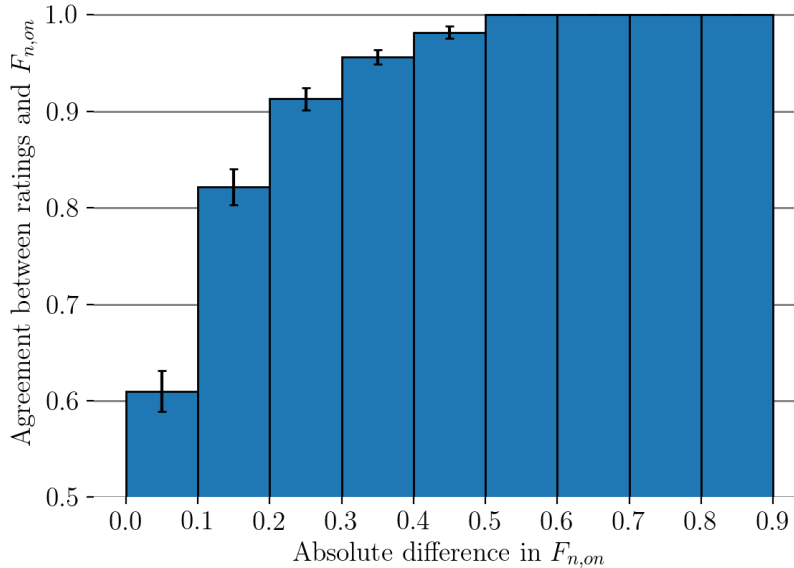


Figure 6.9: Percentage of agreement depending on the difference in  $F_{n,On}$  between the two options, computed on confident answers only.

tend to agree with the F-measure: out of the 315 questions confidently rated by four raters, all four raters agreed with  $F_{n,On}$  for 280 of them (88%), one rater disagreed with  $F_{n,On}$  for 23 of them (7.3%), two raters disagreed with  $F_{n,On}$  for 3 of them (0.95%), three raters disagreed with  $F_{n,On}$  for 3 of them (0.93%), and all raters disagreed with  $F_{n,On}$  for 6 of them (1.9%). Again, this should be considered an upper bound, as it concerns only questions with four confident answers.

We run a linear mixed effect analysis using the amount of agreement between raters as dependent variable (2 if all four raters agree, 1 if one rater disagrees with the other 3, and 0 in the case of a draw), only on the subset of confident answers, and keeping only the questions with four confident answers. We use as dependent variables the difference of  $F_{n,On}$  between the two systems ( $\Delta F$ ), the best  $F_{n,On}$  of the pair ( $F_{best}$ ), the average and standard deviation of the Gold-MSI scores of the four raters for each question (Gold-MSI<sub>avg</sub> and Gold-MSI<sub>std</sub> respectively), and the average reported difficulty (Difficulty<sub>avg</sub>). The resulting coefficients and associated p-values are given in Table 6.5.

Once again, we observe that the bigger the difference in  $F_{n,On}$ , the higher the agreement among raters. However, this time, the  $F_{n,On}$  of the best solution does not seem to have a significant effect (noting that we also have many fewer data

| Feature                   | Coefficient | P-value |
|---------------------------|-------------|---------|
| $\Delta F$                | 0.496       | <0.001  |
| $F_{best}$                | -0.092      | 0.423   |
| Gold-MSI <sub>avg</sub>   | -0.071      | 0.004   |
| Gold-MSI <sub>std</sub>   | -0.016      | 0.778   |
| Difficulty <sub>avg</sub> | -0.176      | 0.003   |

Table 6.5: Coefficients and p-values for the linear mixed effects model using agreement among raters as dependent variable and features as fixed effects.

points). Raters also tend to disagree more with each other when the reported difficulty is higher on average. It also appears that when raters have a high average Gold-MSI, they tend to disagree more with each other. This could be due to the fact that trained musicians might favour different aspects of music (rhythm rather than melody for instance) when making a choice. Disparity in Gold-MSI among raters has no significant effect on whether they agree.

### 6.3.7 Discussion

It appears that the best correlation with ratings is achieved for much higher tolerance thresholds than what is usually used for transcription system evaluations, both for  $F_{n,On}$  and  $F_{n,OnOff}$ . This suggests people are generally relatively forgiving with respect to onset precision, and probably focus on other aspects of music than just onset and offset precision to make their choices. Moreover, the OnOff-Note metric, presented as the most perceptually-relevant evaluation metric by Hawthorne et al. [2018], is actually not the best metric in terms of agreement with human ratings, at least in the case of piano music. On-Note metrics should be favoured, though this may relate to the focus on piano which generally has very salient onsets, but less clear offsets, especially for long notes. OnOff-Note metrics are still useful to assess the performance of a system, as they represent a meaningful objective that is difficult to achieve, but they are not the most representative indicator of the perceptual quality of a transcription system.

Figure 6.9 also shows that when the difference in  $F_{n,On}$  is smaller than 10%, raters confidently disagree with  $F_{n,On}$  as to which transcription is best nearly 40% of the time. This means that in these cases,  $F_{n,On}$  should not be considered as a good descriptor of the quality of a transcription, at least from a perceptual point of view. This is particularly worrying, as very often, differences between systems are of the order of a few percentage points. On the other hand, we compare short segments, which means that a few errors could influence greatly  $F_{n,On}$ , while AMT systems are often compared over hours-long datasets. Also,

in these difficult cases, raters tend to disagree more with each other, so personal judgement also comes into play. In summary, however, the majority of the previous analysis seems to indicate that  $F_{n,O_n}$  is a good enough metric in clear-cut cases where the differences in performance are large, but should probably be treated with caution for small differences between AMT systems.

## 6.4 Defining a new metric

Given the relatively low agreement between ratings and current evaluation metrics, in particular in borderline cases, we propose to define a new evaluation metric, based on the ratings. The general idea is to compute a set of musical features on pairs (AMT output, target), and then train a classifier to output a value between 0 and 1 for each pair based on these features, using the ratings as training data.

### 6.4.1 Comments from participants

We first consider feedback from participants. Out of all participants, twelve left comments related to their decision-making strategies. The melody was mentioned as important in nine comments, making it the most important aspect according to comments, followed by rhythmic aspects (beat/meter/tempo, eight mentions) and harmony (four mentions). Some comments also mentioned higher level, less clearly defined aspects of music: three comments mentioned the “overall impression” was most important, two comments mentioned the presence of major artefacts or out-of-key notes. Overall, three comments mentioned explicitly that the presence of errors was not important as long as other aspects of the music were preserved, and most comments mentioned combinations of the above factors.

### 6.4.2 Feature description

From the previous comments, we define several features to capture various aspects of music, as well as typical AMT mistakes. In the following, we provide high-level definitions for each of these features. Full definitions can be found in Appendix B. We omit the consonance metrics described in Section B.3.10. Indeed, we observed in previous experiments that they did not help our system. Since they are not newly-proposed, for the sake of simplicity, we exclude them from the rest of this section.

### Mistakes in highest and lowest voice

We use the highest and lowest voice of a piece as a proxy for the melody and the bassline, respectively. We define these metrics both framewise and notewise. For highest voice metrics, we define true positives and false negatives as notes in the highest voice of the target that have been correctly detected or missed (respectively). We count as a false positive any extra note that is above the highest voice in the target. From these values, we compute  $P$ ,  $R$ , and  $F$  as described in Section 2.6.2. The lowest voice metrics are defined similarly. To better capture the score rather than the audio signal, we define the highest and lowest voices on targets without taking the pedal into account, while the pedal is otherwise used in the computation of  $F_f$ ,  $F_{n,On}$  and  $F_{n,OnOff}$ .

### Loudness of false negatives

We assume that missing a note that was loud in the original piece is more salient than missing a quiet one, as louder notes are easier to hear, and quiet notes can be masked if played at the same time as louder notes. We define two corresponding metrics:

- Average false negative loudness: the average MIDI velocity of false negatives. Each MIDI velocity is normalised by the average velocity in the ground truth in a two second window centred on the false negative onset.
- False negative loudness ratio: the average ratio between the loudness of false negatives and the maximum loudness of active notes at the time of the false negative onset. We take into account the decay of long notes when computing the maximum loudness at the time of the onset.

### Out-of-key false positives

We assume that out-of-key extra notes are much more noticeable than in-key ones. Instead of relying on key annotations, we define the key of a piece as the set of pitch classes that are active more than 10% of the time. The threshold of 10% is defined heuristically. This definition shows its limits when there are key modulations. We also define a non-binary key-disagreement as the proportion of the time that a pitch class is inactive. We then define two sets of metrics:

- Binary out-of-key: We count the number of false positives whose pitch is out-of-key. We then compute the proportion of out-of-key false positives among false positives, and among all notes in the output.

- Non-binary out-of-key: we compute the average key-disagreement of false positives, and the ratio between the sum of key-disagreements of false positives and the sum of key-disagreements of all detected notes.

### Repeated and merged notes

A common type of mistake in AMT is to have repeated (i.e. fragmented) notes, or incorrectly merged notes. We count as a repeated note any false positive that overlaps with a ground-truth note of the same pitch for at least 80% of its duration, and is preceded by at least one note of the same pitch that overlaps with the same ground-truth note. Conversely, we count as a merged note any false negative that overlaps for at least 80% of its duration with a detected note of the same pitch and is preceded by at least one note of the same pitch that overlaps with the same detected note. In both cases, we compute the proportion of mistakes among all false positives, and among all detected notes.

### Specific pitch mistakes

It is also fairly common to have false positives in specific pitch intervals compared to ground-truth notes: semitone errors (neighbouring notes), octave errors (first partial), and 19 semitone errors (second partial). For these types of mistakes, we define both framewise and notewise metrics, for a given number of semitones  $n_s$  (here  $n_s \in \{1, 12, 19\}$ ).

For framewise metrics, we count a specific pitch false positive for any false positive such that there is a ground truth note  $n_s$  semitones above or below. For notewise metrics, we count a specific pitch false positive for any false positive that overlaps for at least 80% of its duration with a ground truth note  $n_s$  semitones above or below. For  $n_s = 19$ , we only consider ground truth notes 19 semitones below, as second partial mistakes usually only happen 19 semitones above the ground truth. In both cases, we compute the proportion of mistakes among all false positives, and among all detected notes.

### Polyphony level difference

We assume that a mistake is more salient when it is the only note being played and that it will also be noticeable if only a few notes of a big chord are transcribed. To account for this, we compute the absolute difference in polyphony level between the target and the output, at each timestep. We then use the mean, standard deviation, minimum and maximum values of this time series as features.

### Rhythm histogram spectral flatness

Rhythm is another important aspect of music according to raters. We thus define a metric to account for rhythmic imprecision as follows. We first compute the inter-onset interval (IOI) sequence of the output and the target. We keep simultaneous onsets, resulting in an IOI of 0. We then compute a histogram of the IOI values, with bin size of 10ms for IOIs below 100ms, and 100ms from 100ms to 2s (we drop IOIs above that value). This histogram should be more peaky for quantised MIDI files than outputs with rhythm imprecision. To describe this quantitatively, we compute the log-spectral flatness [Johnston, 1988] of both histograms (output and target). We use as a feature the spectral flatness of the output histogram, and the difference in spectral flatness between the output and target histograms.

### Rhythm dispersion

We also propose another approach to characterising rhythm quality, based on K-means clustering [Bishop, 2006] of the IOI set. The general idea is to first run K-means clustering on the target IOIs, and then run K-means clustering on the output IOIs using the cluster centres of the target as initial values. We then compute the distance between cluster centres for the target and the output, as well as the relative difference in standard deviation within each cluster. We use as features the mean, maximum and minimum values across clusters.

Choosing the number of clusters is necessarily heuristic. We determine the number of clusters by computing an IOI histogram as described in 6.4.2, but with wider bins, and choosing the peaks of that histogram as initial values for target IOI clustering.

## 6.4.3 Model fitting

Eventually, we aim to obtain a model that, given a set of features for a pair (AMT output, target), will output a scalar between 0 and 1. The main difficulty is that in our dataset, we do not have such absolute ratings, we only have pairwise comparison ratings. To achieve our goal, we draw inspiration from the contrastive loss approach [Hadsell et al., 2006]. The original contrastive loss is defined as follows: given two inputs  $x_1$  and  $x_2$ , a model  $f$  and a variable  $y$  such that  $y = 1$  if  $x_1$  and  $x_2$  are considered similar,  $y = 0$  otherwise:

$$L = y * |f(x_1) - f(x_2)|^2 + (1 - y) \max(\alpha - |f(x_1) - f(x_2)|, 0)^2 \quad (6.1)$$

In other words, if  $x_1$  and  $x_2$  are similar, the loss tries to bring their outputs together, and if they are dissimilar, it tries to push them apart. The  $\alpha$  parameter

is called the margin: if the distance between  $f(x_1)$  and  $f(x_2)$  is already greater than  $\alpha$ , they are not moved further.

Given a target  $T$ , and two transcriptions of that target  $O_1$  and  $O_2$ , we have, in place of  $x_1$  and  $x_2$ ,  $g(T, O_1)$  corresponding to the set of features computed on  $T$  and  $O_1$ , and  $g(T, O_2)$ , the set of features computed on  $T$  and  $O_2$ . In our ratings, all transcriptions are dissimilar, so  $y$  is always equal to 0. Also, we do not only want  $f(g(T, O_1))$  and  $f(g(T, O_2))$  to be different, we also care about their order. We thus introduce a new variable  $z$  such that  $z = 0$  if  $O_1$  was chosen by the rater, and  $z = 1$  if  $O_2$  was chosen. We want to have  $f(g(T, O_1)) > f(g(T, O_2))$  if  $z = 0$ , and the other way around if  $z = 1$ . We thus define our loss function as:

$$L = \max(\alpha - z * (f(x_2) - f(x_1)) - (1 - z)(f(x_1) - f(x_2)), 0)^2 \quad (6.2)$$

We incorporate the difficulty ratings in the margin: when ratings are confident, we use a higher margin. In practice, we use  $\alpha = 0.5$  when Difficulty = 1, and decrease it by 0.1 for each difficulty level, until  $\alpha = 0.1$  when Difficulty = 5.

We choose to use a simple model, allowing for interpretability of its parameters. Indeed, we want our metric to fit perceptual ratings, but also to serve as a diagnosis tool, allowing us to easily investigate the contribution of each feature in the end result. For that reason, we use logistic regression, using as input all the above-defined features, in addition to the benchmark metrics.

#### 6.4.4 Experiments

##### Setup

We use as input data to the logistic regression model the above features, along with the benchmark metrics defined in Section 2.6.2. We split our dataset between training, validation and test set using a 90%-5%-5% partition, and use 20-fold cross-validation. The splits are made so there is no overlap in targets between the three subsets. There can be some overlap in terms of raters, which means that there is a possibility that the model learns the preferences of some specific participants. Our main concern is that the model should generalise to unseen input, so we still keep these ratings. In each fold, the data is z-normalised (mean=0 and variance=1). The weights of the logistic regression are all initialised to 0. The model is then trained using the Adam optimiser [Kingma and Ba, 2015] with a learning rate of 0.01 for a total of 3000 batches with a batch size of 100, which in practice is enough to ensure convergence. The parameters that achieve the lowest loss on the validation set are then used for testing. In each fold, we train 100 versions of the model (training a model takes about 15s),

to account for potential variation in performance due to the randomness of the training process. We test whether our model agrees with ratings significantly better than  $F_{n,On}$  by running an independent-samples T-test on each fold, and then testing whether the resulting T-values are significantly different from 0. We use 20 folds to have more data points when running the second test, and thus better statistical power in our results.

We focus the evaluation of our models on confident ratings. We thus compute the proportion of agreement between the output of our model and the confident ratings only, i.e. with Difficulty=1 or 2 (notated  $A_{conf}$ ).

### Results and ablation study

All results averaged across folds are shown in Figure 6.10. The dotted line corresponds to  $A_{conf}$  for  $F_{n,On}$ .

First, we train our model using all metrics. We manage to improve slightly (1%) but significantly ( $p < 10^{-6}$ ) the agreement with the ratings, which is encouraging. It has to be noted that the model we used is very simple, and that more sophisticated models should be able to improve even further, though it may not be easy to achieve this without deteriorating interpretability.

In what follows, we investigate feature importance. One approach would be to inspect the weights of the trained logistic regression. However, it might happen that one feature has a high weight in a given model, but when removing it, its absence can be compensated by combinations of other features without decreasing performance. We thus favour an ablation approach to study how essential features are to model ratings, removing groups of features from the feature set and re-training our model as described above. Table 6.6 summarises the configurations we investigate.

Three configurations perform significantly worse than All: NoFeatures, No-Frame-wise, and NoRhythm. Besides, NoFeatures is the only configuration that does not perform significantly better than  $F_{n,On}$  ( $p = 0.33$ ), which shows the usefulness of the feature set we have proposed. The low performance of NoRhythm compared to All shows the importance of the rhythm descriptors we used. This is somewhat contradictory with results from Section 6.3.3: we found that high tolerance thresholds for onsets and offsets gave better agreement, which seemed to indicate that temporal aspects are not important to raters. We suggest that our rhythm descriptor better captures higher-level aspects of rhythm reported as important to raters, such as the presence of a steady pulse and meter, rather than onset precision of individual notes. The fact that No-Frame-wise performs significantly worse than All shows that while  $F_f$  is indeed less correlated to ratings than  $F_{n,On}$ , some frame-wise metrics are useful and

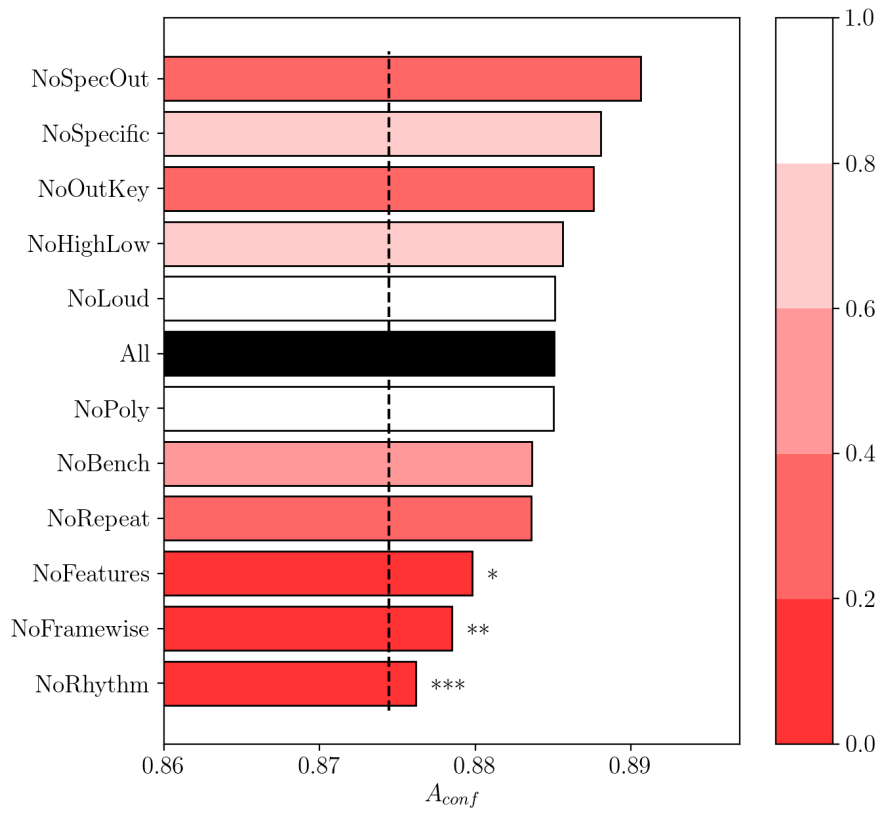


Figure 6.10:  $A_{conf}$  measure for each tested configuration, averaged across folds. The dotted line represents  $A_{conf}$  for  $F_{n,On}$ . Colors represent the  $p$ -value when testing whether each metric is different from the "All" configuration. Asterisks represent results significantly different from All (\*:  $p < 0.1$ , \*\*:  $p < 0.05$ , \*\*\*:  $p < 0.01$  ).

complementary to notewise metrics in modelling the ratings.

On the other hand, it appears that NoHighLow is not significantly worse than All. Yet, melody was the musical aspect that was most mentioned in user comments. We hypothesise that the reason this is not reflected in feature importance is that for the vast majority of examples in our dataset, the highest voice notewise F-measure, which best describes how well the melody was transcribed, is equal to 1. The model probably learns to give a low importance to that feature, as it is often constant. Another hypothesis is that our skyline approach to define the melody and the bassline might not correspond to perception. In the future, we might have to rely for instance on automatic melody estimation methods for symbolic music to better represent the melody.

Interestingly, it appears that some of the metrics we designed, in particular the out-of-key false positives and specific pitch errors, are actually counter-productive: removing them increases  $A_{conf}$ , although with relatively low significance ( $p = 0.40$  and  $p = 0.76$  respectively). We hypothesise that this is due to the definition of these metrics. For instance, if there are no specific pitch mistakes, this could either mean that there were no false positives (which is good), or there were a lot of false positives, none of which corresponded to a specific pitch (which is bad). This could lead to an interaction between specific pitch mistakes and benchmark precision metrics (e.g. penalise low specific pitch and low precision, but not low specific pitch and high precision). The same can be said of out-of-key false positives. However, such interactions cannot be represented by our model (simple logistic regression without interaction terms). As a result, out-of-key and specific pitch mistakes end up distracting the model more than they help. When removing both of these metrics (NoSpecOut configuration), our model reaches an  $A_{conf}$  of 89.1%. Removing other features that have either no impact or a negative impact on  $A_{conf}$  marginally decreases  $A_{conf}$  compared to NoSpecOut.

We make a pre-trained version of our metric available for future use (NoSpecOut configuration). We train it using all the data, without keeping out a validation or test set. Experiments show that in practice, the model does not overfit the training set: the training and validation losses are similar. We thus choose as final parameters those that minimise the loss over the whole training set. Given that we do not keep a held-out test set, we cannot report test performance of this specific released model.

## 6.5 Discussion

In this study, we presented a listening test to rate pairs of AMT systems. We compared perceptual ratings to results given by benchmark evaluation metrics.

| Configuration | Removed features   |
|---------------|--|
| All           | None   |
| NoBench       | Benchmark metrics  |
| NoFeatures    | All features, except benchmark metrics   |
| NoHighLow     | Mistakes in highest and lowest voice   |
| NoLoud        | Loudness of false negatives  |
| NoOutKey      | Out-of-key false positives   |
| NoRepeat      | Repeated and merged notes  |
| NoSpecific    | Specific pitch mistakes  |
| NoPoly        | Polyphony level difference   |
| NoRhythm      | Rhythm histogram spectral flatness and rhythm dispersion   |
| NoFramewise   | Framewise benchmark metrics, framewise highest and lowest voice mistakes, framewise specific pitch errors, polyphony level difference, consonance measures |
| NoSpecout     | Specific pitch mistakes and out-of-key false positives   |

Table 6.6: Description of each tested feature configuration.

We have seen that most of the time, ratings agree with benchmark evaluation metrics, but in some cases (when both transcriptions have low  $F_{n,On}$ , and when the difference in  $F_{n,On}$  between the two transcriptions is low), the agreement greatly decreases. We have proposed new quantitative measures describing musical features, and used them to define a new metric, that agrees with ratings significantly better than  $F_{n,On}$ . We also provide greater insight into which features were important to raters through an ablation study, illustrating in particular the importance of rhythm-related aspects.

Various aspects of this study could be improved. One of the most important would be to try more sophisticated models (e.g., artificial neural networks) to define a new metric. Indeed, the current approach only brings marginal improvement in  $A_{conf}$  compared to  $F_{n,On}$ , some more involved approaches could improve further agreement with ratings. In particular, it would be theoretically possible to define a metric without using handcrafted features, directly by feeding the target and output into the system, but this approach would require more ratings to be trained robustly, and would lack interpretability. Still, some of the features might not have a linear influence on the quality of the transcription, and some may interact. Incorporating such factors into a model may improve performance. We chose a simple but interpretable logistic regression, which

allowed us to verify easily the contribution of each metric to the final score.

Moreover, although we believe that absolute similarity rating between two excerpts is a difficult and ill-defined task [Allan et al., 2007, Flexer and Grill, 2016], it could be interesting to develop a listening test based on absolute similarity ratings between a reference and a single transcription. Provided inter-rater agreement is high enough, it would be interesting to train a regression model to approximate these ratings, and compare the results to those obtained with the current ranking paradigm.

Deeper investigation of the reasons for disagreement between ratings and  $F_{n,On}$  would also be useful to motivate the creation of new metrics. One way to investigate this would be to reproduce the above ablation study, but with a model trained and tested exclusively on ratings that disagree with  $F_{n,On}$ , although the lack of data could make it difficult to achieve significant results, requiring collection of further ratings.

The generalisability of the metric we have designed should also be investigated. First, this metric was only designed for Western classical piano music. It would be interesting to investigate the extent to which it could be applied to other genres (e.g. jazz, non-Western music) and other instruments (e.g. guitar, multi-instrument ensembles). The protocol presented above could be applied with different stimuli to design metrics for other contexts, and potentially define a unified metric that works in every situation. But even in the context of Western classical piano music, some further experiments would have to be run to test the generalisability of our metric. In particular, this metric was trained only on short segments; it remains to be seen whether it scales properly to longer pieces. One way to test our metric would be to run another similar listening test, once again, using pairwise comparisons, but choosing specific, potentially artificial stimuli, to investigate specific points of disagreement: for instance, pairs of examples where our metric and  $F_{n,On}$  disagree as to which is best. By choosing representative examples with the specific aim of comparing these two metrics, much less data would be needed to validate which metric is most closely correlated to human perception.

Finally, this metric was designed to reflect perceptual similarity between the AMT output and the target. Such an evaluation criterion might not be relevant for every application. It is important when the overall musical quality of the transcription matters more than precise transcription of every note, for instance in the context of music creation and production (e.g. quick dictation of musical ideas) or tasks such as automatic accompaniment or cover detection. However, it might not be relevant in cases such as music education, where exact transcription of every note is paramount to properly assess the mistakes made by a student. In this case, reaching an  $F_{n,OnOff}$  of 1 should be the main

objective, regardless of how the transcription sounds. In that regard, our metric complements the usual benchmark metrics to reflect perceptual quality of AMT outputs, but does not replace them.

## 6.6 Reproducibility

To allow further study of the data collected, we make it fully available, along with the stimuli, and the locations in seconds of the manually-selected cut points at the following address: <https://zenodo.org/record/3746863>

We also provide the code of the website at the following address:  
[https://github.com/adrienycart/AMT\\_perception\\_website](https://github.com/adrienycart/AMT_perception_website)

A Python implementation of the used features, and the pre-trained metric can be found at the following address: <https://github.com/adrienycart/PEAMT>

## Chapter 7

# Conclusions and Perspectives

### 7.1 Summary

In this thesis, we have investigated music language modelling using neural networks. We have applied these models in various contexts.

First, in Chapter 3, we have looked into polyphonic music prediction using LSTMs. By investigating the performance of our model with various timesteps, we have shown in Section 3.3 the importance of the choice of the input representation, and the shortcomings of current metrics, that cannot describe the qualitative differences we observed. We proposed new metrics for polyphonic music sequence prediction in Section 3.4.2, and combined them into a parametric training loss. We investigated the influence of the training loss parameters, both for polyphonic music sequence prediction (intrinsic evaluation, Section 3.5.3) and using our model as an MLM for AMT (extrinsic evaluation, Section 3.5.4), showing that the loss parameters influence the performance of our model in terms of  $P_f$ ,  $R_f$ ,  $P_{n,O_n}$  and  $R_{n,O_n}$  in consistent patterns across timesteps. Moreover, tuning these parameters appropriately resulted in an improvement in AMT performance without changing the complexity of our model.

We have also investigated two types of methods to improve AMT, namely transduction methods in Chapter 4 and MLM decoding methods in Chapter 5. For transduction, we have investigated various transduction model designs, in particular binary neurons and the F-measure training loss, with various input acoustic models. We showed in Section 4.3 that using as post-processing a CNN transduction model with binary outputs trained to optimise the F-measure significantly improved AMT performance compared to simpler post-processing

procedures. We have further shown in Section 4.3.6 that while quantising the outputs to a 16th-note grid always improves results, using a 16th-note transduction processing step gives either similar or better results than using a smaller, time-constant timestep in the transduction process.

For MLM decoding, we have proposed in Section 5.2 an approach to dynamically combine the predictions from the acoustic and language models using an additional blending model, showing a significant improvement over using a static linear combination of the two predictions. We have also shown in Section 5.4.2 that using a musically-related timestep for the MLM significantly improves AMT performance, beyond the fact that the outputs are quantised. We have further shown that using automatically-detected beat positions instead of ground truth annotations to compute the musically-related timesteps still yielded a significant improvement over time-constant timesteps. Besides, we have shown in Section 5.4.4 that using scheduled sampling makes the MLM more robust during inference, improving the AMT results when the acoustic model gives noisy predictions. Overall, the proposed decoding method improved over simple thresholding and smoothing methods by more than 8% in terms of the onset-only notewise F-measure, and improved by 5% over a strong newly-proposed HMM baseline, without using any extra ground-truth information. Besides this performance improvement, we have published augmented annotations for the MAPS dataset (described in Appendix A), in the form of updated MIDI files containing key and rhythm information such as the key signatures, time signatures, and tempo curve for each piece.

We have further looked into the perceptual relevance of benchmark AMT evaluation metrics in Chapter 6. We have collected perceptual judgements of AMT quality through an online listening test, and showed that while the current evaluation metrics agree to a great extent with participant ratings in clear-cut cases, the agreement decreases in the difficult cases (*i.e.* when the difference in metrics is smaller). We have proposed a new evaluation metric, based on a simple linear model trained to fit participant ratings using handcrafted, musically-motivated features (described in detail in Appendix B), and showed that our approach correlates with ratings significantly better than benchmark metrics.

While some progress was made in these directions, there are still many avenues to explore for future research on MLMs for AMT. Besides the improvements to the presented experiments that we discussed in each chapter, there are some more general directions that would be worth investigating. In what follows, we discuss some limitations of the work we conducted, and propose directions for improvement.

## 7.2 Limitations and Perspectives

### 7.2.1 Stylistic aspects

The work conducted in this thesis focuses on Western classical tonal piano music. This choice was partly motivated by the availability of big datasets. It remains to be seen how well the models trained on this type of data can generalise for instance to other styles of Western piano music, such as jazz or popular music. This would give us insights about how much of the specific style of music was modelled by our approach.

Besides, it would be interesting to investigate the same approaches but trained on different kinds of data (other styles, non-Western music, multi-instrument ensembles), and see how much of the proposed techniques can be applied to other instruments, styles, and cultures. Beyond the type of data used for training, doing so would give us important information about the extent to which the models and methods presented in this thesis can be generalised beyond the specific use case of Western classical piano music.

### 7.2.2 Note-based MLMs

In this thesis, we have investigated various input representations. We call these representations frame-based: although a timestep might correspond to time segments of varying duration, they are still sequences of fixed-length vectors representing simultaneous pitch information, similar to frames of a spectrogram. We focused on such frame-based MLMs, as we believe they allow a simpler, more straightforward integration with frame-based acoustic models. Throughout this thesis, we mentioned various limitations of this representation.

A first limitation is the fact that when using a binary piano-roll representation, it is impossible to distinguish repeated and held notes. We mentioned in Sections 3.6 and 5.5 a ternary representation, that would allow us to do so, without exploring it further. Another limitation is the inherent tradeoff between temporal precision and modelling of time dependencies: when using smaller timesteps, the representation contains more repeated frames, which makes it more difficult for the model to learn time-dependencies on longer timescales. We explored solutions to improve that tradeoff, for instance using a loss function giving more weight to transitions in Chapter 3, or using musically-related timesteps to use longer timesteps while keeping a sufficient temporal precision in Chapters 4 and 5, but this remains a strong limitation of frame-based representations.

Beyond frame-based representations, other ways of representing music sequences could be explored, in particular note-based representations, where each

sequence element represents a note event, rather than a vector of pitches active at a given time. Such a representation allows circumventing the problem of repetition in frames. Moreover, they allow explicit modelling of onsets and offsets of notes, and make it possible to distinguish repeated and held notes.

Various models using note-based representations have been proposed for music generation. The Performance RNN [Oore et al., 2018] uses a representation similar to MIDI, where each token is a message such as *note on*, *note off*, or *time shift*. The MuseNet [Payne, 2019] also uses a note-based representation, where each token encodes pitch, volume, and instrument information, with extra tokens for time delay. These approaches can provide useful inspiration to develop note-based MLMs.

One difficulty of note-based MLMs is that there is not a 1-to-1 correspondence between the MLM processing step and the acoustic model processing step. Some solutions can be found in the ASR literature: in ASR, characters or words do not follow a 1-to-1 correspondence with audio frames. The Connectionist Temporal Classification loss [Graves et al., 2006] is a common solution for that problem in ASR, by using as loss the sum of the likelihoods for all possible alignment paths. While this would work for monophonic music (it was used for instance in [Román et al., 2018]), this might be more challenging to adapt to polyphonic music. In speech and in monophonic music, monotonicity can be assumed in the alignment, meaning that the symbols in the output sequence come in the same order as in the input. For polyphonic music, the fact that voices overlap means that monotonicity depends on the encoding, and is not guaranteed.

### 7.2.3 Self-similarity

The approach we have explored in this thesis is based on the analysis of a corpus. A dataset of music is analysed, and the patterns found in this dataset are then used to make predictions on new music. This approach allows us to learn general musical features, related to music theory, such as for instance some sense of tonality or meter. However, it is limited when it comes to making precise predictions as to what continuations might come after a sequence of notes: it is very unlikely that a new piece of music is similar enough to a piece of music in the analysed dataset to be able to make such predictions. Moreover, the memory of an LSTM is actually quite limited in time: it is able to recognise and predict patterns on the scale of a few notes, but would not retain them on longer timescales (*e.g.* throughout a minutes-long music piece).

On the other hand, music is very often strongly self-similar: repetition is an important characteristic of music, be it in note and rhythm patterns, or in whole

sections of a piece. This self-similarity can be exploited to make predictions. Simple systems such as [de Reuse, 2019] that try to match the longest identical sequence in the past and use the continuation of this sequence as prediction for the future, obtain very good results in music prediction tasks. Other more involved methods, that allow for slight modifications of the repetitions, such as [Walder and Kim, 2018], are also promising, although this method was only applied to simple music sequences that do not take rhythm into account.

An interesting middle-ground approach was proposed in [Medeot et al., 2018] for monophonic music generation. This approach uses an RNN trained on music data, similarly to this thesis, with an additional network, that learns repetition patterns in the dataset. This repetition network is then used to bias the outputs of the first towards repeating some parts of the previous outputs (either the pitches, the durations, or both, with various periodicities), much like using  $\mathcal{H}_{pp}$  as loss in Section 3.5.3 biased the network towards outputting more in-key false positives. This is a promising approach, as it allows us to benefit both from corpus-learned regularities, and repetitions in the specific piece being analysed.

#### 7.2.4 Higher-level musical rules

In this thesis, we have mainly used as MLMs simple LSTM networks. This model is suitable for our definition of a language model, *i.e.* a system that is able to assess the likelihood of an output sequence based on prior information. While we have seen that LSTMs are relatively successful at modelling some statistical regularities found in music sequence data, and can indeed improve empirical results in various instances, they should not be considered as a good model of music. Indeed, while music can be seen as sequential data from which statistical regularities can be extracted, it is also made of complex interactions, both within each voice, and between voices. These interactions are often of logical nature rather than statistical, and can be found at many hierarchical levels. Although powerful statistical models could potentially learn some of these interactions, a basic LSTM surely cannot.

It would be interesting to investigate better ways to model such interactions. One option would be to look into models that combine rule-based and data-based approaches, such as for instance [Jaques et al., 2017], described in Section 2.4.3. In order to specifically encode the hierarchical structure of music, the Hierarchical Multiscale RNN proposed in [Chung et al., 2017] for character-level language modelling can be considered. This system stacks several RNNs, updated at different rates at each level. In [Chung et al., 2017], 3 such layers are stacked, the lowest corresponding to characters, the middle one to word units, and the highest to prepositions. Although it would still probably not model the

vertical dependencies in music in all their complexity, this hierarchical structure is very well-suited to the temporal organisation of music. The lower-level could correspond to the frame-level, then a higher level would represent beats, then bars, then maybe even sections of the piece. Such a hierarchical structure was successfully used for music generation [Roberts et al., 2018], in a variational auto-encoder setup where the decoder is a hierarchical RNN operating at the bar and 16th note level. However, each level is updated with fixed periodicity, while in [Chung et al., 2017], boundaries can happen after an arbitrary number of steps to allow for tempo and structure flexibility. Some extra regularisation constraints could probably be added in order to make boundaries approximately regularly spaced (or exactly in the case of beats in a bar), to fit this architecture to the specific problem of music modelling.

### 7.2.5 MLMs for Complete Music Transcription

In this thesis, we have focused on descriptive music transcription, *i.e.* trying to describe as accurately as possible what was played in a specific performance. In particular, the output of our systems can be represented as a list of note, or an unquantised MIDI file. This can be useful for many applications, from music education to further MIR tasks such as cover song detection (see also Section 7.2.7). However, these representations cannot be easily read by musicians, who are more used to working with music scores. Complete music transcription would be an important task to investigate to allow the users to be able to directly read the transcribed outputs.

For CMT, there are many aspects of music that can be exploited. While the methods we investigated in this thesis were mostly focused on the note content, CMT is composed of many other tasks, such as rhythm quantisation, pitch spelling, meter and key detection, that would benefit from the use of MLMs. Various approaches have been proposed for conversion of MIDI into score, one of the most comprehensive examples being [Temperley, 2009]. Some approaches have been proposed for CMT of piano music from audio [Nakamura et al., 2018], using some musical prior knowledge in particular in stream separation of the left and right hand parts. It would be interesting to investigate more approaches that allow us to post-process the output of an acoustic model into score-format directly, with the use of prior musical knowledge.

In CMT, we make a distinction between semantic information and typesetting information, that we define as follows. Semantic information is related to note pitches and durations. It includes multi-pitch detection, and rhythm transcription. Typesetting information is related to how the semantic information is written in a score format. It includes key and time signatures, streaming

separation, rhythm spelling and pitch spelling (it is controversial to include the key and time signature as typesetting information, as one might argue that they carry important semantic information about the piece, but we still count them as typesetting information as the exact same sequence of pitches and durations could be represented in different ways using different key or time signatures). MLMs could be used to improve transcription of these two categories of information. The work presented in this thesis is related to inference of semantic information, but only focuses on the note content (i.e. pitch, onset and offset times of notes in seconds). Including rhythm quantisation in the same process would be interesting, and we believe that the work we conducted using musically-relevant timesteps is a useful intermediate step to that end. For typesetting tasks, MLMs would be even more important, as in these cases, the decisions are less driven by observations from the acoustic model, and rely more on musical rules and prior knowledge, as well as detected semantic information. Obviously, having a joint MLM for all these tasks would be better, but also very challenging given the size of the resulting search space.

### 7.2.6 Transduction vs. end-to-end training

The transduction approach we have proposed is trained using the same data as AMT systems. One could thus imagine, instead of training the acoustic and transduction models in two separate steps, to train them jointly, so they can both adapt to each other. In particular, in the case of neural-network based acoustic models, training a transduction model jointly with the acoustic model would be possible, and would boil down to adding more layers to the acoustic model.

We believe that the approach explored in Chapter 4 is more general, as it is compatible with any type of acoustic model, not only with neural networks. For instance, it can be used with a PLCA acoustic model, as in Section 4.2. Moreover, compared to end-to-end training, our approach uses an extra training objective: both the output of the acoustic model and the output of the transduction model try to approximate the target piano roll. It would be interesting to train an end-to-end model combining an acoustic model and a transduction model using these two training objectives in a multi-task training fashion [Zhang and Yang, 2017], and investigate how the performance differs from the same architecture simply stacking the two models and using a single training objective.

### 7.2.7 On the usefulness of MLMs for AMT

The work conducted in this thesis aims at improving fully-automatic music transcription performance. It makes various assumptions. One of them is that the music to transcribe is conforming to a certain musical style (in our case, Western tonal piano music, see also Section 7.2.1). MLMs are thus useful when the style being transcribed is well defined. Moreover, we have seen in Chapter 5 that MLMs are particularly useful when the acoustic model performs poorly. This would make them particularly suitable to transcribe for instance historical recordings, where the musical style can often be known in advance and recording conditions are usually noisier.

However, this assumption might be counter-productive in certain application cases, in particular in music education. In this case, the aim is to detect what notes students play as accurately as possible. In particular, what has to be transcribed might contain some errors, such as out-of-key notes. In this case, we do not want these errors to be removed by a potential MLM: although they are unlikely in an actual piece of music, such errors are essential to assess the performance of a student. Using an MLM is thus not advisable in such a case. It has to be noted that in the context of music education, the score is often provided. This additional information can help detecting correctly played notes and mistakes more easily, but the problem, usually called score-informed transcription [Benetos et al., 2012], is then slightly different from the one investigated in this thesis.

It is also possible that the use of an MLM might be counter-productive when transcribing music that does not conform to the style it was designed for. For instance, some harmonies are very uncommon in classical music, but very common in jazz. These would then be considered as mistakes by an MLM trained on classical music, and thus taken out, while they might have been fully intended by the performer, and rightly detected by the acoustic model. The risk is then that the transcription outputs are too biased towards a specific style, and end up being believable from a musical point of view, but different from what was performed. In such cases, models that use little to no musical priors, and mostly focus on acoustic modelling of musical sounds, such as e.g. [Hawthorne et al., 2019] or [Kelz et al., 2019], might be more appropriate.

Another assumption is that the user cannot, or will not be involved in the transcription process. This assumption is not valid in what we call end-user AMT. Under this umbrella, we group use cases such as music creation and production (for instance quick dictation of musical ideas) or musicology (for instance analysis of jazz and other improvised music), and any other use case where a user will read and use the output transcription. In particular, this does

not include cases where the output is further used as a feature for other tasks, such as cover detection or automatic accompaniment. In a lot of end-user AMT cases, the user is skilled enough to transcribe the input themselves, they just want a tool to make the process quicker and easier. In particular, they would be able to spot, and potentially correct transcription mistakes. While the methods proposed in this thesis could benefit end-user AMT, they focus on an objective that is not necessarily the most relevant: rather than trying to be as accurate as possible, such systems should try to make the transcription process as quick as possible for the user. A study was run on that question [Holzapfel and Benetos, 2019], showing that it was not significantly quicker to use automatic transcription outputs as starting point than to start from scratch (although for non-experts, it was non-significantly quicker and reported as significantly easier). Semi-automatic methods that take into account user feedback to refine the transcription output (using active learning techniques for instance [Settles, 2011]), and that use MLMs for automatic completion or suggestion (similar to autocorrection in NLP) could improve the speed and easiness of the transcription process. We believe that is an under-explored research direction in the literature, including in this thesis, but nonetheless important and worth exploring.

### 7.2.8 Application-specific AMT evaluation

In Chapter 6, we proposed a new evaluation framework for AMT, based on perceived polyphonic music similarity. This metric is based on perceptual data that we gathered without putting restrictions on the level of musical training of the raters, or on how confident they were in their judgement. One might argue that it would be more relevant to only use ratings by trained musicians, as their judgement should be trusted more than non-musicians. We did not put such restrictions, as we believe that all answers are useful, that most people are able to perceive and rate the similarity between some music excerpts, and that our metric should summarise all opinions, both from musicians and non-musicians. Still, it would be interesting to see to what extent the results would differ when using only answers given by experts.

Moreover, we only used stimuli for a specific style and instrument: Western tonal classical piano music. This limits the scope of application of this metric; it should not be assumed that all musical aspects would have the same importance for other styles (for instance, jazz, popular music, or non-Western music) and for other instruments (for instance guitar or multi-instrument ensembles). It would be interesting to replicate this study, using the same protocol and musical features, with stimuli from different genres and instruments. Not only would we obtain new metrics that can be applied in these cases, but we could also com-

pare feature importance for these various contexts, and study to which extent the results obtained in Chapter 6 are style- or instrument-specific. Generally speaking, it would also be interesting to investigate the perceptual validity of evaluation metrics in various MIR tasks where it might be relevant, such as for instance chord transcription, automatic drums transcription, or melody transcription. Listening tests could be conducted for these tasks using a similar protocol, and new metrics could be defined similarly, although a different set of features would have to be defined for each of these tasks.

It has to be noted that beyond the specific style, instrument, or task to which it is applied, an evaluation function based on the perceptual similarity between the AMT output and the target might not be the most appropriate for all AMT use cases. For instance, as discussed in Section 7.2.7, for music education, a system should aim for an error rate of 0%, regardless of how the target or the transcribed output sound. For end-user AMT, a better way to evaluate such a system would be how easy or how quick it is to obtain the correct transcription starting from this AMT output. Other methods, such as for instance approaches based on the edit distance, might be more appropriate in this case. Finally, when AMT is used for other tasks, such as cover detection, it is important to evaluate the impact of AMT performance on the performance of the whole system. When comparing MLM intrinsic performance and AMT performance in Chapter 3, we observed that there was little correlation between MLM cross entropy and AMT  $F_{n,O_n}$ . Similarly, it should not be assumed that some improvement in AMT  $F_{n,O_n}$  would necessarily result in some improvement in the systems where AMT is used.

### 7.3 Conclusion

Throughout this thesis, we investigated some higher-level research questions, beyond the engineering goal of improving the accuracy of AMT systems.

One of them is the influence of the timestep used for the analysis, and in particular, the use of musically-relevant timesteps rather than time-constant timesteps. This proved to be an important factor in the performance of systems in various contexts. In Chapter 3, we showed that for polyphonic music prediction, the input representation greatly influences the behaviour of a model: using a tempo-normalised timestep allows the model to predict when note transitions might happen, and using a shorter timestep biases the network towards repeating the previous observations. In Chapter 4, we showed that in the context of polyphonic music sequence transduction, some improvements could be obtained by using a musically-relevant timestep analysis, depending on the transduction model and acoustic model. In Chapter 5, we showed that, for MLM decoding,

operating the MLM using a musically-relevant timestep could greatly improve AMT performance, and that this improvement is still significant when using automatically-detected beat positions. Given all this evidence, we can say that the input timestep indeed has a strong influence on the behaviour and performance of a system, and that taking into account musical considerations in the design of a system can greatly improve their performance.

This work also questioned the suitability of the evaluation metrics used in the context of music language modelling and AMT. In Chapter 3, we have investigated the correlation between the intrinsic and extrinsic performance of an MLM. We have demonstrated that the usual intrinsic evaluation metric for MLMs, the cross entropy, is actually not correlated to extrinsic performance in AMT. We have proposed various evaluation metrics, that allow us to get more qualitative insights into the behaviour of a given model, and shown that these evaluation metrics were related to some AMT metrics. In Chapter 6, we have assessed the correlation between benchmark AMT evaluation metrics and human perception of the quality of AMT, formulated as the perceptual similarity between the AMT output and the target. We have gathered perceptual data on this question through listening tests, that we make available for future research. We have shown that although in clear-cut cases, these metrics match human judgement, the agreement decreases in more difficult cases. We have proposed new, musically-grounded evaluation metrics, and we have combined them using perceptual ratings into a new evaluation metric, which is significantly better correlated to human judgement than benchmark metrics. Still, this improvement was quite modest, and it remains to be seen whether this method generalises to new data. But beyond this specific proposed solution, we hope this work will help to initiate a discussion about the perceptual relevance of AMT evaluation metrics and lay out a common framework to investigate this question.

Beyond these specific questions, this thesis aimed to investigate whether MLMs are worth using at all for AMT. The results presented in Section 5.4 show that, indeed, a substantial performance improvement can be obtained when using MLMs for AMT. However, this improvement was only observed in quite specific conditions: when transcribing perfect performances of classical piano music. MLMs could be counter-productive when transcribing excerpts that do not conform to a specific well-identified style, for instance in the context of music creation, or in music education, where the performances might contain mistakes (as discussed in Section 7.2.7). Moreover, this improvement was made at the expense of computational efficiency. The approach we presented is theoretically causal, with only a small necessary time-delay of one beat when using beat-related timesteps (although in our experiments, the beat-tracking method is not causal). On the other hand, the search process using the MLM

is computationally-costly. Our approach is thus best suited for use cases where computational cost is not a sensitive issue, excluding for instance real-time or embedded applications.

## Appendix A

# A-MAPS: Augmented MAPS Dataset with Rhythm and Key Annotations

The MAPS dataset has become a widely-used benchmark dataset for AMT. It contains around 18 hours of classical piano music, along with aligned MIDI transcriptions. However, the information it contains is limited to the pitch, onset and offset times in seconds of the notes in the recording, and sustain pedal activation (velocity is also included, but rarely used). In this appendix, we describe the A-MAPS dataset, a new set of ground-truth MIDI files for the MIDI Aligned Piano Sounds (MAPS) dataset [Emiya et al., 2010]. This dataset was presented in the Late Breaking and Demos session at ISMIR 2018 (item 6 in Section 1.4).

In this thesis, we have proposed various methods that incorporate additional musical information into AMT systems, such as meter. With the progress of AMT, Complete Music Transcription (CMT, see Chapter 1) has also gained attention [Nakamura et al., 2018]. To make evaluation and comparison with state-of-the-art AMT systems possible, we propose augmented annotations for MAPS, that include in particular rhythm (meter, note values), key, as well as other general information, all the while preserving the pitch, onset and offset as they were in the original MAPS files. We make these annotations available for further use<sup>1</sup>, in the form of MIDI files.

---

<sup>1</sup><http://c4dm.eecs.qmul.ac.uk/ycart/a-maps.html>

## A.1 Method

As described in Section 2.7.2, the MAPS dataset contains MIDI files of polyphonic piano music, along with aligned audio renditions, generated using synthetic pianos and Disklavier acoustic pianos. It also contains recordings of isolated notes and chords, that we do not consider here.

The MAPS MIDI files used to render the audio were taken from the Piano-Midi.de<sup>2</sup> (PM) database. The PM database was made by manually editing the velocities and the tempo curves of quantised MIDI files in order to give them a natural interpretation. These MIDI files originally contained much more information (tempo and note values, but also key, textual indications, sustain pedal...) which, unfortunately, was not kept in MAPS. We aim to retrieve this additional information and include it in the MAPS ground-truth MIDI files.

Directly using the PM files as ground truth is not possible, since this dataset is continuously updated by its creator, meaning many files have been slightly modified since the creation of MAPS. We thus resort to a symbolic MIDI-to-MIDI alignment method [Nakamura et al., 2017] to align pairs of files. From these alignments, we get a correspondence table that tells us to which PM quantised point each MAPS note is aligned. More precisely, we get a table  $T$  such that  $T[i] = [t, q]$  where  $t$  is a time in seconds and  $q$  is a quantised time in quarter notes. We remove the duplicates, so that each  $q$  is unique (when a single  $q$  value corresponds to many  $t$  values, we keep the first  $t$  value). From this correspondence table, we get a tempo curve following closely the MAPS files. Let  $bpm(t)$  be the tempo value in quarter notes per minute at time  $t$ . For each  $T[i] = [t_1, q_1]$ ,  $T[i + 1] = [t_2, q_2]$  we have the formula:

$$bpm(t) = \frac{q_2 - q_1}{t_2 - t_1} \times 60 \text{ for } t \in [t_1, t_2[$$

In some cases, the ordering is not preserved, *i.e.*  $q_1 < q_2$  but  $t_1 \geq t_2$ . In such cases, we delete one of the two problematic lines, keeping the one for which the quantised time was closest to a 16th note subdivision. When the two quantised times are both on a 16th note subdivision, we keep the one that results in the smallest tempo deviation.

Adapting the tempo curve does not yield perfect results: some notes were modified in the PM files after the creation of MAPS, or displaced during the re-alignment process in the making of MAPS. To get the exact notes from MAPS, we match the notes in the MAPS files with those in the tempo-aligned PM files using the `mir_eval` toolbox [Raffel et al., 2014]. We then remove all the PM notes that are unmatched, and add the MAPS notes that are unmatched.

---

<sup>2</sup><http://piano-midi.de/>

| $\delta_{mean}$ |       |      | $\delta_{max}$ |     |     |
|-----------------|-------|------|----------------|-----|-----|
| min             | med   | max  | min            | med | max |
| 0.0             | 0.028 | 3.18 | 0              | 1   | 384 |

Table A.1: Evaluation of the beat annotations: minimum, median and maximum values in MIDI ticks of  $\delta_{mean}$  and  $\delta_{max}$  across the dataset

We add the missing notes to the staff holding the PM note to which they were aligned by the alignment algorithm [Nakamura et al., 2017]. When adding a MAPS note aligned to no PM note, we add it to the right hand staff if its MIDI pitch is above 60, to the left hand staff otherwise (this concerns 2.7% of the notes). Eventually, all the correct notes are retrieved, and their onset and offset values are within 5ms of the original.

Because of this second step, some notes in the aligned files are not exactly quantised to beat subdivisions, although they might have been in the original PM files. To know how much, we compare the quantised versions of the original PM files and aligned files. We match the notes of the two files with [Nakamura et al., 2017], and for each pair of notes, we compute the deviation between their onsets in MIDI ticks (1 tick is a 480<sup>th</sup> of a quarter note). Then, for each file, we compute the mean ( $\delta_{mean}$ ) and maximum ( $\delta_{max}$ ) deviations. We report the minimum, median, and maximum values of  $\delta_{mean}$  and  $\delta_{max}$  across the dataset in Table A.1. It has to be noted that the large maximum values recorded are most likely due to notes that were modified in later versions of the PM MIDI files.

## A.2 Contents

All the annotations that were in the original PM files were retrieved. They include:

- ✓ Tempo curve
- ✓ Time signature
- ✓ Durations of notes in fraction of a quarter note (some of them are approximate)
- ✓ Key signature (always written as the major relative)
- ✓ Separate left and right hand staff
- ✓ Text annotations from the score (tempo indications, coda...).

Yet, they do not include all the information needed for CMT, mostly due to limitations of the MIDI format. In particular, they do not include:

- ✗ Pitch spelling: C♯ is the same as D♭
- ✗ Rhythm spelling: a quarter note tied to an eighth note is the same as a dotted quarter note
- ✗ Ornament notation: trills are written as a succession of notes, not as a separate symbol
- ✗ Dynamics information: only the velocity of individual notes is available
- ✗ Short key modulations: only changes in key signature are written.

### A.3 Applications

These annotations can be useful in the development of musically-informed AMT systems (for instance, key- or meter-dependent systems). They can be used directly as input to the systems to test their performance in the ideal case, or to assess the performance of sub-modules of the system.

They can also be used for other tasks than AMT, such as beat tracking, and meter or key detection from audio. Although the note values given in the annotations are not always exact, they can provide some useful estimates for rhythm quantisation. The textual annotations could also be used for structure analysis.

Finally, even though some elements are missing to reconstruct a full score, we hope that all the information contained in these annotations can help working towards CMT, by providing a big amount of annotated audio data and allowing comparison with previous AMT systems.

## Appendix B

# Musical Features for Automatic Music Transcription Evaluation

### B.1 Introduction

Automatic Music Transcription (AMT) is most often evaluated with metrics that penalise all mistakes equally (see [Bay et al., 2009] for a description of common AMT metrics). However, all mistakes are not equally salient to human listeners: for instance, out-of-key false positives tend to be very noticeable. In order to get more insights into the types of mistakes made by a system, we define new, lower-level metrics that can be computed on pairs (target, AMT output). We aim to define these metrics so that they capture musical aspects that are reported as important by human raters, and mistakes commonly made by AMT systems.

These metrics are defined with polyphonic piano music transcription as the main application. However, most of them are more general and can be applied in other contexts with minor modifications.

In what follows, we describe the notations we use throughout this document in Section B.2.

### B.2 Data format

The AMT output and target are usually represented either as a piano roll or a list of notes.

A piano roll is a  $N_p \times T$  binary matrix, where  $T$  corresponds to the number of timesteps, and  $N_p$  to the number of considered pitches (88 in the case of piano). We notate the estimated and ground-truth piano rolls as  $\hat{M}$  and  $M$  respectively. Roughly,  $M[p, t] = 1$  if and only if pitch  $p$  is active at timestep  $t$ . We notate  $M_t$  the binary  $N_p$ -vector, which corresponds to the  $t$ -th frame of  $M$ .

The AMT output and target can also be represented as a list of notes. We notate them  $\hat{N}$  and  $N$  respectively. They are lists of potentially different lengths  $\hat{n}$  and  $n$  respectively. Each element of  $N$  is a tuple  $(s, e, p, v)$  where  $s$  and  $e$  are the start and end times,  $p$  is the MIDI pitch, and  $v$  is the original velocity of the note. Each element of  $\hat{N}$  is a tuple  $(s, e, p)$  (same, without velocity, as it is often not estimated by AMT systems).

## B.3 Features

### B.3.1 Benchmark Framewise Precision, Recall, F-measure

These framewise metrics are computed on piano-roll outputs, with a frame duration of 10ms. We notate for each frame  $t$  the number of true positives, false positives and false negatives as respectively,  $TP(t)$ ,  $FP(t)$  and  $FN(t)$ . A true positive is counted when  $M[p, t] = 1$  and  $\hat{M}[p, t] = 1$ . We define the framewise Precision, Recall and F-measure as:

$$\mathcal{P} = \frac{\sum_{t=0}^T TP(t)}{\sum_{t=0}^T TP(t) + FP(t)} \quad (\text{B.1})$$

$$\mathcal{R} = \frac{\sum_{t=0}^T TP(t)}{\sum_{t=0}^T TP(t) + FN(t)} \quad (\text{B.2})$$

$$\mathcal{F} = \frac{2 \cdot P \cdot R}{P + R} \quad (\text{B.3})$$

### B.3.2 Benchmark Onset-only Notewise Precision, Recall, F-measure

These notewise metrics are computed on lists of note outputs. We count a true positive when a note  $(\hat{s}, \hat{e}, \hat{p})$  is detected and there is a reference note  $(s, n, p)$  such that:

$$|\hat{s} - s| < 50ms \wedge \hat{p} = p \quad (\text{B.4})$$

Each reference note must be matched to at most one estimated note.

We then compute the Precision, Recall and F-Measure as above. The maximum matching between target and output is computed using `mir_eval`<sup>1</sup>.

In what follows, we use this definition to determine whether a detected note is a true positive, unless stated otherwise, as it was shown in Chapter 6 that onset-only F-measure is the benchmark evaluation metric that correlates best with human perception (in the case of piano). When used for other instruments, the onset-offset definition might be preferred.

### B.3.3 Benchmark Onset-offset Notewise Precision, Recall, F-measure

These metrics are the same as above, with the difference that a true positive is counted when:

$$|\hat{s} - s| < 50ms \wedge \hat{p} = p \wedge |\hat{e} - e| < \max(50ms, 0.2 * (e - s)) \quad (\text{B.5})$$

We then compute Precision, Recall and F-Measure as above.

### B.3.4 Number of mistakes in highest and lowest voice

The general idea is that very often, mistakes in the melody or the bassline are more salient than mistakes in middle voices. We use the highest and lowest voice as a proxy for the melody and the bassline, respectively. The highest and lowest voices can be defined both framewise or notewise.

Usually, the ground truth is defined so that it corresponds as much as possible to what is contained in the audio signal. In particular, when the sustain pedal is used, the offsets of notes are usually extended to correspond to the actual duration for which they sound. When defining the highest and lowest voice, we choose to not take the sustain pedal into account, in order to stick as much as possible to the original partition, and avoid excessive overlapping of notes. We assume that this is accessible, as most of the time, the sustain pedal is given as an external MIDI control-change parameter, that can either be taken into account or not.

#### Frame-wise

The highest voice  $H_M$  is a time series such that:

$$H_M(t) = \max_{p \in [0, N_p]} (p \mid M[p, t] = 1) \quad (\text{B.6})$$

---

<sup>1</sup>[https://github.com/craffel/mir\\_eval](https://github.com/craffel/mir_eval)

If  $M_t$  is all zeros, we define by default  $H_M(t) = -1$ .

- A true positive is counted for each  $(p, t)$  such that  $p = H_M(t)$  and  $H_M(t) \neq -1$  and  $\hat{M}[p, t] = 1$ .
- A false negative is counted for each  $(p, t)$  such that  $p = H_M(t)$  and  $H_M(t) \neq -1$  and  $\hat{M}[p, t] = 0$ .
- A false positive is counted for each  $(p, t)$  such that  $\hat{M}[p, t] = 1$  and  $p > H_M(t)$ . We count all false positives above the highest pitch in the target.

With these definitions, we can compute Precision, Recall, and F-measure as before.

The lowest voice can be defined similarly.

### Notewise

The highest voice  $H_N$  is a list of notes such that for all  $(s, e, p)$  in  $H_N$ :

$$\exists t \in [s, e] \mid \forall (s', e', p') \in N, t \notin [s', e'] \vee p' < p \quad (\text{B.7})$$

In other words, it is the set of notes that are the highest sounding note at some point in time. In order to account for cases when, for instance, a chord is slightly arpeggiated and a middle note is played before the highest note, we include a minimum duration  $d_H$  for which the note has to be the highest sounding one:

$$\exists t_1, t_2 \in [s, e] \mid t_2 - t_1 > d_H \wedge \forall (s', e', p') \in N, [t_1, t_2] \cap [s', e'] = \emptyset \vee p' < p \quad (\text{B.8})$$

- A true positive is counted for each note that is a true positive and that is matched to a note in  $H_N$ .
- A false negative is counted for each note in  $H_N$  that is left unmatched.
- A false positive is counted for each note  $(s, e, p)$  in  $\hat{N}$  that is left unmatched and such that

$$\exists t \in ]s, e[, \forall (s', e', p') \in N, t \notin ]s', e'[\vee p' < p$$

or with a threshold:

$$\exists t_1, t_2 \in [s, e] \mid t_2 - t_1 > d_H \wedge \forall (s', e', p') \in N, [t_1, t_2] \cap [s', e'] = \emptyset \vee p' < p$$

We count all false positives above the highest pitch in the target.

We set  $d_H = 0.5$ , this value is set heuristically, in accordance with the usual threshold for benchmark notewise metrics.

With these definitions, we can compute Precision, Recall, and F-measure as before.

The lowest voice can be defined similarly.

### B.3.5 Loudness of false negatives

The idea is that a missed note will be less noticed if it was played very softly in the input than if it was very loud originally. It will be even less noticed if there are some louder notes played at the same time.

Similar metrics could be defined with false positives, but since most current AMT systems do not estimate note velocities, we do not take them into account.

We define two such metrics: the normalised false negative loudness, and the false negative loudness ratio.

#### Normalised false negative loudness

For each false negative, we normalise its loudness by the average loudness in a 2-seconds window centered on the onset of the false negative. In other words, for a given false negative  $(s, e, p, v)$  in  $N$  define  $V$  as:

$$V = \{(s', e', p', v') \mid |s - s'| < d_L\} \tag{B.9}$$

where  $d_L$  is set as 1 second. we then compute the normalised loudness as:

$$\text{NormLoud} = \frac{v \times |V|}{\sum_{(s', e', p', v') \in V} v'} \tag{B.10}$$

where  $|V|$  denotes the cardinal of  $V$ . We then compute the average normalised loudness for all false positives. It has to be noted that  $V$  is never empty; it always contains at least  $(s, e, p, v)$ . When  $(s, e, p, v)$  is the only note in  $V$ , the ratio is equal to 1.

#### False negative loudness ratio

We also compute the ratio between the missed note velocity and the loudest note sounding in the ground truth at the time of the attack of the missed note. This ratio is necessarily smaller than 1. In particular, it is equal to 1 when the missed note is the only one at that time. We could then average this ratio for all the false negatives.

To do so, we assume an exponential decay of the amplitude of notes, to reflect cases where a loud note is held while other notes are played. We use

an exponential decay rate, applied to the velocities directly. We stop the decay after 1 second, to avoid notes fading out completely. Given  $a(p)$  the decay rate of a note given its pitch, we define the time varying velocity  $v(t)$  of a note  $(s,e,p,v)$  as:

$$v(t) = \begin{cases} ve^{-a(p)(t-s)} & \text{if } t \in [s, s + 1] \\ ve^{-a(p)} & \text{if } t \in [s + 1, e] \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.11})$$

The decay rate is obtained from the values presented in Figure B.1, taken from [Cheng, 2016]. For each pitch, we average the decay rates across velocities (this has little influence for lower pitches). Then, we do a linear regression on the averaged decay rates, in order to have smoother, less piano-dependent decay rates estimates. Using that, we can then compute the velocity of notes through time. The formula we obtain for  $a(p)$  is:

$$a(p) = 0.050532 + 0.021292 * p \quad (\text{B.12})$$

The loudness ratio of a false negative  $(s, e, p, v)$  is then defined as:

$$\text{LoudRatio} = \frac{v}{\max_{(s',e',p',v') \in N, t \in [s-d_R, s+d_R]} v'(t)} \quad (\text{B.13})$$

where  $d_R$  is a parameter that we set to 50ms.

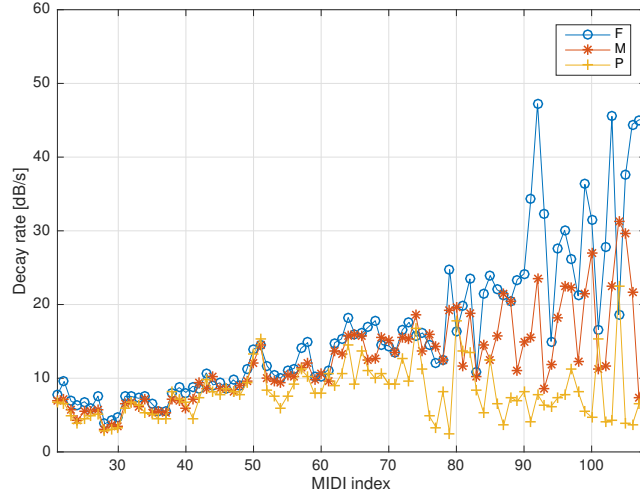


Figure B.1: Decay rates per note, for various velocities (taken from [Cheng, 2016]).

### B.3.6 Out-of-key false positives

The idea is that out-of-key inserted notes are particularly salient. However, we do not want to rely on key annotations to evaluate that. Instead, we rely on a pitch profile that is automatically computed from the target. We propose two versions of this metric, one based on a binary pitch profile, and one on a non-binary pitch profile.

The following definitions assume that the tonality is constant throughout the considered music excerpt. Behaviour is undefined if that is not the case.

#### Binary pitch profile

We define the binary pitch profile as the set of pitch classes that are active more than 10% of the time in the target. This threshold is set heuristically. Generally-speaking, the higher threshold, the more notes will be considered as out-of-key. It remains to be seen to what extent varying this threshold influences the metrics.

More precisely, we define an active-pitch-class time series  $A_p(t)$  such that:

$$A_p(t) = 1 \iff \exists(s, e, q) \in N \mid q \equiv p \pmod{12} \wedge t \in [s, e] \quad (\text{B.14})$$

$A_p(t)$  is defined using the target without pedal, similarly to Section B.3.4.

We write  $P_b$  the set of in-key pitches such that:

$$p \in P_b \iff p \equiv q \pmod{12} \wedge \frac{1}{T} \sum_{t \in [0, T[} A_q(t) > 0.1 \quad (\text{B.15})$$

We write  $FP_{\hat{N}}$  the total number of false positive notes in  $\hat{N}$ ,  $FP_{P_b, \hat{N}}$  the number of out-of-key false positives (i.e. the false positives such that  $p \notin P_b$ ), and  $|\hat{N}|$  the total number of notes in  $\hat{N}$ . We then define two ratios:

$$\mathcal{O}_{b,t} = \frac{FP_{P_b, \hat{N}}}{|\hat{N}|} \text{ and } \mathcal{O}_{b,p} = \frac{FP_{P_b, \hat{N}}}{FP_{\hat{N}}} \quad (\text{B.16})$$

$\mathcal{O}_{b,t}$  is the proportion of out-of-key mistakes among all detected notes (and is thus correlated with notewise recall), while  $\mathcal{O}_{b,p}$  is the proportion of out-of-key mistakes among all false positives.

### Non-Binary pitch profile

For each pitch class  $q$ , we define its fitness to the tonality  $F(q)$  as the proportion of the time this pitch class is active:

$$F(q) = \frac{1}{T} \sum_{t \in [0, T[} A_q(t) \quad (\text{B.17})$$

Then, we define the key-disagreement of a false positive  $(s, e, p)$  as:  $1 - F(q)$  for  $q$  such that  $q \equiv p \pmod{12}$

We then compute 2 values:  $\mathcal{O}_{b,t}$  the average key-disagreement of false positives divided by the average key-disagreement of all detected notes, and  $\mathcal{O}_{b,p}$  the un-normalised average key-disagreement of false positives.

### B.3.7 Specific Pitch Errors

A common type of AMT mistake is to have false positives in specific pitch intervals compared to ground-truth notes: semitone errors (neighbouring notes), octave errors (first partial), and 19 semitone errors (second partial). We define errors for specific pitches  $p_e$  with  $p_e \in \{1, 12, 19\}$ . We define these metrics both framewise and notewise.

#### Framewise

$\hat{M}[p, t]$  is a specific pitch error if and only if the following conditions are all true:

- $\hat{M}[p, t] = 1$
- $M[p, t] = 0$
- $M[p - p_e, t] = 1 \vee M[p + p_e, t] = 1$
- $\forall i \in [t - \delta, t[, M[p, i] = 0$

where  $\delta$  is a parameter that we set heuristically to 50ms, in accordance with the threshold used for benchmark notewise metrics. The last condition is to ensure that erroneous continuations of correct notes are not penalised if there was also a target note  $p_e$  semitones apart right after.

For  $p_e = 19$ , we only consider ground-truth notes 19 semitones below, as second partial mistakes usually only happen above the ground truth notes. The third condition becomes simply:  $M[p - p_e, t] = 1$

We then compute two different ratios: let  $N_s$  be the number of specific pitch errors,  $N_f$  the number of frames, and  $N_e$  the total number of false positives:

$$\mathcal{S}_{p_e, f, t} = \frac{N_s}{N_f} \text{ and } \mathcal{S}_{p_e, f, p} = \frac{N_s}{N_e} \quad (\text{B.18})$$

The first is correlated to  $\mathcal{P}$ , while the second can be biased when  $\mathcal{P}$  is very high (an output with only 1 error that happens to be a specific pitch error will have  $\mathcal{S}_{f,p} = 1$ )

### Notewise

A note  $(s, e, p)$  is a specific pitch false positive if:

- $(s, e, p)$  is a false positive
- $\exists (s', e', p') \in N \mid |p - p'| = p_e \wedge \frac{\min(e, e') - \max(s, s')}{e - s} > 0.8$

The last condition boils down to saying that there is a ground truth note  $p_e$  semitones apart from  $(s, e, p)$  that overlaps with  $(s, e, p)$  for 80% of  $(s, e, p)$ 's duration. The value 80% was set heuristically, and echoes the benchmark onset-offset notewise metrics condition requiring that the offset of a note is within 20% of its duration.

For  $p_e = 19$ , we only consider ground truth notes 19 semitones below  $(s, e, p)$ , for the same reason as above.

Similarly as framewise metrics, we compute 2 ratios: the proportion of specific pitch mistakes among all detected notes, and among false positives.

### B.3.8 Repeated and merged notes

Another common type of mistake in AMT is to have repeated (i.e. fragmented) notes, or incorrectly merged notes.

#### Repeated notes

A note  $(s, e, p) \in \hat{N}$  is counted as a repeated note when it fulfills the following conditions:

- It is an Onset-only false positive
- $\exists (s', e', p') \in N \mid p = p' \wedge \frac{\min(e, e') - \max(s, s')}{e - s} > 0.8$
- $\exists (s'', e'', p'') \in \hat{N}$  such that:
  - $(s'', e'', p'') \neq (s, e, p)$
  - $p'' = p'$
  - $\frac{\min(e'', e') - \max(s'', s')}{e'' - s''} > 0.8$
  - $e'' < s$

Put more simply, a note is considered as a repeated note if it is a false positive, if it overlaps with a ground-truth note, and if there is another previous detected note that overlaps with the same ground-truth note.

We can then compute either the proportion of repeated notes among false positives, or among all notes, as with the previous metrics. Once again, for both of these metrics, the threshold 80% was set heuristically, and echoes the benchmark onset-offset notewise metrics condition requiring that the offset of a note is within 20% of its duration.

### Merged notes

A note  $(s, e, p) \in N$  is counted as a merged note when it fulfills the following conditions:

- It is an Onset-only false negative
- $\exists(s', e', p') \in \hat{N} \mid p = p' \wedge \frac{\min(e, e') - \max(s, s')}{e - s} > 0.8$
- $\exists(s'', e'', p'') \in N$  such that:
  - $(s'', e'', p'') \neq (s, e, p)$
  - $p'' = p'$
  - $\frac{\min(e'', e') - \max(s'', s')}{e'' - s''} > 0.8$
  - $e'' < s$

A note is thus considered as a merged note if it is a false negative, if it overlaps with a detected note, and if there is another previous ground-truth note that overlaps with the same detected note.

We can then compute either the proportion of merged notes among false negatives, or among all notes, as with the previous metrics.

## B.3.9 Rhythm features

### Rhythm histogram spectral flatness

Rhythm is an important aspect of music. We thus define a metric to account for rhythmic imprecision as follows. We define  $O$  and  $\hat{O}$  the (ordered) list of note onsets of  $N$  and  $\hat{N}$  respectively, potentially with repetition.  $O$  is of same size as  $N$ .

Based on these, we compute inter-onset-intervals ( $IOI$  and  $I\hat{O}I$ ) as the first derivative of  $O$  and  $\hat{O}$  respectively. Let  $n$  be the number of notes:

$$\forall 0 \leq i < n - 1, IOI(i) = O(i + 1) - O(i) \quad (\text{B.19})$$

We then compute a normalised histogram of the IOIs, with bins as follows: from 0 to 100ms, we use a bin size of 10ms, and from 100ms to 2s, we use a bin size of 100ms. Overall, we have 29 bins. We notate this list  $b$ , and the resulting histogram  $h_r$ . The spacing in bins is set heuristically; however, it could have a strong influence on the result. We leave it to future work to quantify that influence.

From this IOI, we compute its spectral flatness [Johnston, 1988], which is defined as the ratio of the geometric mean of the histogram over its arithmetic mean: It is used usually on power spectra, and represents the peakiness of the spectrum. It is useful in our case, as quantised rhythms would give an IOI histogram with only some non-zero bins, corresponding to specific note values, while rhythm imprecision would spread the values across several bins. We thus have:

$$S_f = \frac{1}{29} \sum_{0 \leq i < 29} \log(h_r[i]) - \log\left(\frac{1}{29} \sum_{0 \leq i < 29} h_r[i]\right) \quad (\text{B.20})$$

In practice, we add  $\epsilon = 10^{-5}$  to deal with 0 values in  $h_r$ .

We then compute the spectral flatness value for  $N$  and  $\hat{N}$ . We use as features the spectral flatness for  $\hat{N}$  and the difference between the spectral flatness of  $\hat{N}$  and  $N$ .

### Rhythm dispersion

Another approach to attempt to characterise rhythmic deviations is to run K-means clustering [Murphy, 2012] on the IOI set. Ideally, each cluster would correspond to one note value, with small variations due to tempo deviations and interpretation mostly. We can then assess the mean and standard deviation within each cluster.

Setting the number of clusters here is a tough problem. If we do not have enough clusters, one cluster might correspond to several note values, which would result in artificially high standard deviation. If we have too many clusters, we might end up with several clusters corresponding to the same note value, or in the extreme case, one cluster per note.

To determine the number of clusters and their initial centers, we compute a normalised IOI histogram on the target, similar to the previous, but with higher bin size: 20ms between 0 and 0.1s, and 200ms between 0.1 and 2s. We then choose as initial cluster centres all the peaks in the resulting  $h_r$ . Here again, the spacing in bins is set heuristically; however, it has to be noted that it might have a strong influence on the number of clusters.

We first run K-means clustering on the target *IOI* set. After convergence, we use the resulting cluster means as initial values to run K-means clustering on

the estimated *IOI* set. We then compute the distance between cluster means for the estimated and target *IOI* sets, and the relative difference between the cluster standard deviations for the estimated and target *IOI* sets. We use as feature the mean, maximum and minimum across clusters, for both the centre drifts and standard deviation differences.

### **Validating Rhythm features**

We have seen in the experiments presented in Chapter 6 that these rhythmic features have a high importance when modelling perceptual ratings of AMT quality. In order to validate that these metrics do capture rhythm deviations, we run some experiments.

We use as target the AMT outputs for all the stimuli in presented in Section 6.2. We use as outputs various modified versions of these same MIDI files, by order of rhythm regularity (high to low):

**Quant-constant:** Quantised MIDI files with 16th note precision, using a constant tempo equal to the average tempo over the whole segment (we use the A-MAPS tempo and beat annotations described in Appendix A);

**Quant:** Quantised MIDI files with 16th note precision, using a time-varying tempo, with the ground-truth 16th note positions;

**Noisy-100:** Add uniform noise in [-100ms,100ms] to the onset times;

**Noisy-300:** Add uniform noise in [-300ms,300ms] to the onset times.

We report in Table B.1 the mean and standard deviation (std) of the rhythm features in each condition. It appears that the features behave generally as expected. The mean spectral flatness is lowest for Quant-constant, and highest for the Noisy configurations. Besides, the spectral flatness difference is negative for the quantised versions, and positive for the noisy versions. For the rhythm dispersion values, we see a similar trend: for quantised versions, the average change in std is negative, while it is positive for noisy versions. Moreover, the greater the noise, the greater the average change in std.

However, it appears that increasing the noise level does not change the mean spectral flatness of the outputs, which is kind of surprising. This might be due to the bin size we used: since we use bins of size 100ms between 100ms and 2s, small differences in noise might be hard to catch. Another possibility is that since we use short examples, in a lot of cases, histogram bins contain one single value. Adding noise to that value will change the bin it is counted in, but will not change the overall spectral flatness. This might also explain why the dispersion average drift also increases with the noise level: although the noise

APPENDIX B. MUSICAL FEATURES FOR AMT EVALUATION

| Feature                      | Quant-constant |       | Quant         |       | Noisy-100     |       | Noisy-300    |       |
|------------------------------|----------------|-------|---------------|-------|---------------|-------|--------------|-------|
|                              | mean           | std   | mean          | std   | mean          | std   | mean         | std   |
| Spectral Flatness Output     | <i>-9.842</i>  | 0.857 | -9.572        | 0.909 | <b>-6.695</b> | 0.814 | -6.751       | 0.772 |
| Spectral Flatness Difference | <i>-1.850</i>  | 0.973 | -1.580        | 0.898 | <b>1.298</b>  | 1.217 | 1.241        | 1.380 |
| Dispersion Avg. std Change   | <i>-0.010</i>  | 0.019 | <i>-0.010</i> | 0.017 | 0.024         | 0.020 | <b>0.080</b> | 0.038 |
| Dispersion Min. std Change   | <i>-0.032</i>  | 0.036 | -0.029        | 0.041 | -0.001        | 0.050 | <b>0.037</b> | 0.074 |
| Dispersion Max. std Change   | 0.012          | 0.031 | <i>0.009</i>  | 0.026 | 0.049         | 0.027 | <b>0.125</b> | 0.048 |
| Dispersion Avg. Drift        | <i>0.025</i>   | 0.022 | <i>0.025</i>  | 0.026 | 0.038         | 0.032 | <b>0.129</b> | 0.047 |
| Dispersion Min. Drift        | <i>0.008</i>   | 0.009 | <i>0.008</i>  | 0.009 | 0.015         | 0.013 | <b>0.057</b> | 0.045 |
| Dispersion Max. Drift        | <i>0.046</i>   | 0.050 | <i>0.046</i>  | 0.059 | 0.065         | 0.068 | <b>0.209</b> | 0.095 |

Table B.1: Rhythmic feature means and standard deviation (std) across all stimuli, with 4 levels of rhythmic precision. Highest mean values are in bold, lowest mean values are in italic.

is centred on zero, it is likely applied to many clusters with one single, or few values in it, so the drift does not always cancel out on average within a cluster.

It also appears that the dispersion feature values are very similar for both quantised versions. This might be due to the fact that we use short segments, and that tempo variations are quite small usually, so there is probably little difference between the two quantised conditions.

### B.3.10 Consonance measures

We choose 3 different consonance measures: one based on periodicity/inharmonicity, one based on partials interference, and one based on culture (statistical frequency in a corpus). These are computed using Peter Harrison’s implementation<sup>2</sup>. In particular, we use the following features:

- `hutch_78_roughness` for partials interference
- `har_18_harmonicity` for periodicity
- `har_19_corpus` for culture.

These 3 consonance measures were shown to correlate best with perceptual ratings of consonance [Harrison and Pearce, 2020].

We then compute these consonance measures on the output and target piano rolls, using an *event* timestep: one timestep per new onset or offset. The above features are undefined for silence, we thus do not take them into account in the computations. We then compute the weighted average (using as weight each frame’s duration in sections), the weighted standard deviation, minimum and maximum value for each feature, both on the output and target piano rolls. We use as features the weighted average, the weighted standard deviation, minimum and maximum computed on each consonance measures on the output piano roll.

<sup>2</sup><https://github.com/pmcharrison/incon>

### B.3.11 Polyphony level

We assume that a mistake is more salient when it is the only note being played. Conversely, if a big chord is supposed to be played, but few notes are detected, this will be noticeable.

We compute the difference in polyphony level as a time series:

$$\text{Poly}(t) = \left| \sum_{0 \leq p < 88} \hat{M}[p, t] - \sum_{0 \leq p < 88} M[p, t] \right| \quad (\text{B.21})$$

We then use as features the mean, standard deviation, minimum and maximum of this series.

## B.4 Summary

We provide a table summarising all the features. The first column corresponds to feature groups (as described in the sections above), the second column describes each scalar value that can be found within that feature group, and the last column describes whether higher is better for that metrics: “Yes” if higher is better, “No” if lower is better, “/” when it depends on other factors.

| Feature group                                 | Sub-feature | Higher is better? |
|---|-------------|-------------------|
| Benchmark<br>Framewise metrics                | Precision   | Yes               |
|   | Recall      | Yes               |
|   | F-measure   | Yes               |
| Benchmark<br>Onset-only<br>notewise metrics   | Precision   | Yes               |
|   | Recall      | Yes               |
|   | F-measure   | Yes               |
| Benchmark<br>Onset-Offset<br>notewise metrics | Precision   | Yes               |
|   | Recall      | Yes               |
|   | F-measure   | Yes               |
| Framewise<br>mistakes in highest<br>voice     | Precision   | Yes               |
|   | Recall      | Yes               |
|   | F-measure   | Yes               |
| Framewise<br>mistakes in lowest<br>voice      | Precision   | Yes               |
|   | Recall      | Yes               |
|   | F-measure   | Yes               |
| Notewise mistakes<br>in highest voice         | Precision   | Yes               |
|   | Recall      | Yes               |
|   | F-measure   | Yes               |

*APPENDIX B. MUSICAL FEATURES FOR AMT EVALUATION*

| <b>Feature group</b>                        | <b>Sub-feature</b>   | <b>Higher is better?</b> |
|---|--|--------------------------|
| Notewise mistakes<br>in lowest voice        | Precision  | Yes                      |
|   | Recall   | Yes                      |
|   | F-measure  | Yes                      |
| Loudness                                    | Normalised false negative loudness   | No                       |
|   | False negatives loudness ratio   | No                       |
| Binary out-of-key<br>false positives        | Proportion among false positives   | No                       |
|   | Proportion among detected notes  | No                       |
| Non-binary<br>out-of-key false<br>positives | Average key-disagreement of false positives  | No                       |
|   | Average key-disagreement of false positives normalised by the average key-disagreement of all detected notes | No                       |
| Framewise<br>semitone errors                | Proportion among false positives   | /                        |
|   | Proportion among detected notes  | /                        |
| Framewise octave<br>errors                  | Proportion among false positives   | /                        |
|   | Proportion among detected notes  | /                        |
| Framewise<br>third-harmonic<br>errors       | Proportion among false positives   | /                        |
|   | Proportion among detected notes  | /                        |
| Notewise semitone<br>errors                 | Proportion among false positives   | /                        |
|   | Proportion among detected notes  | /                        |
| Notewise octave<br>errors                   | Proportion among false positives   | /                        |
|   | Proportion among detected notes  | /                        |
| Notewise<br>third-harmonic<br>errors        | Proportion among false positives   | /                        |
|   | Proportion among detected notes  | /                        |
| Repeated notes                              | Proportion among false positives   | /                        |
|   | Proportion among detected notes  | /                        |
| Merged notes                                | Proportion among false positives   | /                        |

*APPENDIX B. MUSICAL FEATURES FOR AMT EVALUATION*

| <b>Feature group</b> | <b>Sub-feature</b>   | <b>Higher is better?</b> |
|----------------------|--|--------------------------|
|                      | Proportion among detected notes                                    | /                        |
| Rhythm histogram     | Value computed on output   | /                        |
| spectral flatness    | Relative difference between value computed on output and on target | /                        |
| Rhythm dispersion    | Mean centre drift  | No                       |
|                      | Minimum centre drift   | No                       |
|                      | Maximum centre drift   | No                       |
|                      | Mean cluster standard deviation difference                         | /                        |
|                      | Minimum cluster standard deviation difference                      | /                        |
|                      | Maximum cluster standard deviation difference                      | /                        |
| Consonance measures  | Mean of <code>hutch_78_roughness</code>                            | /                        |
|                      | Standard deviation of <code>hutch_78_roughness</code>              | /                        |
|                      | Minimum of <code>hutch_78_roughness</code>                         | /                        |
|                      | Maximum of <code>hutch_78_roughness</code>                         | /                        |
|                      | Mean of <code>har_18_harmonicity</code>                            | /                        |
|                      | Standard deviation of <code>har_18_harmonicity</code>              | /                        |
|                      | Minimum of <code>har_18_harmonicity</code>                         | /                        |
|                      | Maximum of <code>har_18_harmonicity</code>                         | /                        |
|                      | Mean of <code>har_19_corpus</code>                                 | /                        |
|                      | Standard deviation of <code>har_19_corpus</code>                   | /                        |
|                      | Minimum of <code>har_19_corpus</code>                              | /                        |
|                      | Maximum of <code>har_19_corpus</code>                              | /                        |

Table B.2: Summary of all the proposed evaluation metrics.

# Bibliography

- Taketo Akama. Controlling symbolic music generation based on concept learning from domain knowledge. In *20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.
- Julien Allali, Pascal Ferraro, Pierre Hanna, and Matthias Robine. Polyphonic alignment algorithms for symbolic music retrieval. In *International Symposium on Computer Music Modeling and Retrieval/International Conference on Auditory Display Joint conference (CMMR/ICAD)*, 2009.
- Hamish Allan, Daniel Müllensiefen, and Geraint A. Wiggins. Methodological considerations in studies of musical similarity. In *8th International Society for Music Information Retrieval Conference (ISMIR)*, 2007.
- Moray Allan and Christopher Williams. Harmonising chorales by probabilistic inference. In *Advances in Neural Information Processing Systems*, 2005.
- Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *34th International Conference on Machine Learning (ICML)*, 2017.
- R Harald Baayen, Douglas J Davidson, and Douglas M Bates. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of memory and language*, 59(4):390–412, 2008.
- Mert Bay, Andreas F. Ehmann, and J. Stephen Downie. Evaluation of multiple-f0 estimation and tracking systems. In *10th International Society for Music Information Retrieval Conference (ISMIR)*, 2009.
- Juan Pablo Bello, Laurent Daudet, and Mark B Sandler. Automatic piano transcription using frequency and time-domain information. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 14(6):2242–2251, 2006.
- Emmanouil Benetos. Polyphonic note and instrument tracking using linear dynamical systems. In *AES International Conference on Semantic Audio*, 2017.

- Emmanouil Benetos and Tillman Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.
- Emmanouil Benetos, Anssi Klapuri, and Simon Dixon. Score-informed transcription for automatic piano tutoring. In *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, 2012.
- Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.
- Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2019.
- Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015.
- Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.
- Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- Emmanuel Bigand, Charles Delbé, Bénédicte Poulin-Charronnat, Marc Leman, and Barbara Tillmann. Empirical evidence for musical syntax processing? Computer simulations reveal the contribution of auditory short-term memory. *Frontiers in Systems Neuroscience*, 8, June 2014. Article number 94.
- Frédéric Bimbot, Emmanuel Deruty, Gabriel Sargent, and Emmanuel Vincent. System & Contrast: A polymorphous model of the inner organization of structural segments within music pieces. *Music Perception: An Interdisciplinary Journal*, 33(5):631–661, 2016.
- Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.
- Rachel M. Bittner and Juan Jose Bosch. Generalized metrics for single-f0 estimation evaluation. In *20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

- Rachel M. Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan Pablo Bello. Deep salience representations for f0 estimation in polyphonic music. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- Sebastian Böck and Markus Schedl. Enhanced beat tracking with context-aware neural networks. In *International Conference Digital Audio Effects (DAFx)*, 2011.
- Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.
- Sebastian Böck, Florian Krebs, and Gerhard Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.
- Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: A new python audio and music signal processing library. In *Proceedings of the 24th ACM International Conference on Multimedia*, 2016a.
- Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016b.
- Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *29th International Conference on Machine Learning (ICML)*, pages 1159–1166, 2012.
- Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. High-dimensional sequence transduction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3178–3182, 2013.
- Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. *Deep learning techniques for music generation*. Springer, 2019.
- Judith C. Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.
- Emre Cakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic

- sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 25(6):1291–1303, 2017.
- Francisco J. Cañadas Quesada, Nicolás Ruiz Reyes, Pedro Vera Candeas, Julio J. Carabias, and Saturnino Maldonado. A multiple-F0 estimation approach based on Gaussian spectral modelling for polyphonic music transcription. *Journal of New Music Research*, 39(1):93–107, 2010.
- Dorian Cazau, Yuancheng Wang, Olivier Adam, Qiao Wang, and Gregory Nuel. Improving note segmentation in automatic piano music transcription systems with a two-state pitch-wise hmm method. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- Zhehuai Chen, Jasha Droppo, Jinyu Li, Wayne Xiong, Zhehuai Chen, Jasha Droppo, Jinyu Li, and Wayne Xiong. Progressive joint modeling in unsupervised single-channel overlapped speech recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(1):184–196, 2018.
- Tian Cheng. *Exploiting Piano Acoustics in Automatic Transcription*. PhD thesis, Queen Mary University of London, 2016.
- Tian Cheng, Matthias Mauch, Emmanouil Benetos, Simon Dixon, et al. An attack/decay model for piano transcription. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- Srikanth Cherla, Son N. Tran, Artur d’Avila Garcez, and Tillman Weyde. Discriminative learning and inference in the recurrent temporal RBM for melody modelling. In *International Joint Conference on Neural Networks (IJCNN)*, 2015.
- Taemin Cho and Juan Pablo Bello. A feature smoothing method for chord recognition using recurrence plots. In *12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011.
- Jan Chorowski and Navdeep Jaitly. Towards better decoding and language model integration in sequence to sequence models. In *18th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2017.
- Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*, 2014.
- Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. In *5th International Conference on Learning Representations (ICLR)*, 2017.

- Andrea Cogliati and Zhiyao Duan. A metric for music notation transcription accuracy. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- Tom Collins, Barbara Tillmann, Frederick S. Barrett, Charles Delbé, and Petr Janata. A combined model of sensory and cognitive representations underlying tonal expectations in music: from audio signals to behavior. *Psychological Review*, 121(1):33–65, 2014.
- Darrell Conklin and Ian H. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.
- Adrien Daniel, Valentin Emiya, and Bertrand David. Perceptually-based evaluation of the errors usually made when automatically transcribing music. In *9th International Society for Music Information Retrieval Conference (ISMIR)*, 2008.
- Manuel Davy, Simon Godsill, and Jerome Idier. Bayesian analysis of polyphonic western tonal music. *The Journal of the Acoustical Society of America*, 119(4):2498–2517, 2006.
- Timothy de Reuse. Copyforward: Point-set matching for predicting patterns. *15th Music Information Retrieval Evaluation eXchange (MIREX)*, 2019.
- Arnaud Dessen, Arshia Cont, and Guillaume Lemaitre. Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In *11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010.
- Hao-Wen Dong and Yi-Hsuan Yang. Convolutional Generative Adversarial Networks with binary neurons for polyphonic music generation. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- Simon Durand, Juan P Bello, Bertrand David, and Gaël Richard. Downbeat tracking with multiple features and deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015.
- Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in Statistics*, pages 569–593. Springer, 1992.
- Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.

- Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with WaveNet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. PMLR, 2017.
- Sebastian Ewert, Meinard Muller, and Peter Grosche. High resolution audio synchronization using chroma onset features. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009.
- Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art*, 2(1):7, Jul 2019.
- Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378, 1971.
- Arthur Flexer and Thomas Grill. The problem of limited inter-rater agreement in modelling music similarity. *Journal of New Music Research (JNMR)*, 45(3):239–251, 2016.
- Klaus Frieler, Dogac Basaran, Frank Höger, Hélène-Camille Crayencour, Geoffrey Peeters, and Simon Dixon. Dont hide in the frames: Note-and pattern-based evaluation of automated melody extraction algorithms. In *6th International Conference on Digital Libraries for Musicology*, 2019.
- Magdalena Fuentes, Brian McFee, Hélène C. Crayencour, Slim Essid, and Juan Pablo Bello. Analysis of common design choices in deep learning systems for downbeat tracking. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- Zoubin Ghahramani. Learning dynamic Bayesian networks. In *International School on Neural Networks, Initiated by IIASS and EMFCSC*, pages 168–197. Springer, 1997.
- Michael Good. MusicXML for notation and analysis. *The virtual score: representation, retrieval, restoration*, 12:113–124, 2001.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

- Masataka Goto. AIST annotation for the RWC music database. In *7th International Society for Music Information Retrieval Conference (ISMIR)*, 2006.
- Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. Rwc music database: Popular, classical and jazz music databases. In *3rd International Society for Music Information Retrieval Conference (ISMIR)*, 2002.
- Elaine Gould. *Behind bars: the definitive guide to music notation*. Faber Music Ltd, 2016.
- Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006.
- Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.
- Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.
- Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.
- Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Polyphonic music analysis database based on gttm. In *2nd Conference on Computer Simulation of Musical Creativity (CSMC)*, 2017a.
- Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. deepgttm-iii: Multi-task learning with grouping and metrical structures. In *International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2017b.
- Peter Harrison and Marcus T. Pearce. Simultaneous consonance in music perception and composition. *Psychological Review*, 127(2):216, 2020.
- Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO

- dataset. In *International Conference on Learning Representations (ICLR)*, 2019.
- Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, abs/1207.0580, 2012a.
- Geoffrey E. Hinton, Nitsh Srivastava, and Kevin Swersky. Neural networks for machine learning. *Coursera, video lectures*, 264, 2012b.
- Keiji Hirata, Satoshi Tojo, and Masatoshi Hamanaka. Implementing “A Generative Theory of Tonal Music”. *Journal of New Music Research*, 35(4):249–277, 2006.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and Development in Information Retrieval*, 1999.
- Andre Holzapfel and Emmanouil Benetos. Automatic music transcription and ethnomusicology: a user study. In *20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.
- Xuedong Huang and Li Deng. An overview of modern speech recognition. In *Handbook of Natural Language Processing*, 2010.
- Andreas Jansson, Eric J. Humphrey, Nicola Montecchio, Rachel M. Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep u-net convolutional networks. In *18th International Society for Music Information Retrieval Conference, (ISMIR)*, 2017.
- Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. Tuning recurrent neural networks with reinforcement learning. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
- James D. Johnston. Transform coding of audio signals using perceptual noise criteria. *IEEE Journal on Selected Areas in Communications*, 6(2):314–323, 1988.
- Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning (ICML)*, 2015.

- Dan Jurafsky and James H. Martin. *Speech and language processing. Vol. 3.* Pearson London, 2014.
- Rudolf E. Kalman. Canonical structure of linear dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 48(4): 596, 1962.
- Hirokazu Kameoka, Kazuki Ochiai, Masahiro Nakano, Masato Tsuchiya, and Shigeki Sagayama. Context-free 2D tree structure model of musical notes for Bayesian modeling of polyphonic spectrograms. In *13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012.
- Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Bock, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. *17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- Rainer Kelz, Sebastian Böck, and Gerhard Widmer. Deep polyphonic adsr piano note transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.
- Filip Korzeniowski and Gerhard Widmer. A fully convolutional deep auditory model for musical chord recognition. In *IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2016.
- Filip Korzeniowski and Gerhard Widmer. On the futility of learning complex frame-level language models for chord recognition. In *AES International Conference on Semantic Audio*, 2017.
- Filip Korzeniowski and Gerhard Widmer. Improved chord recognition by combining duration and harmonic language models. *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- Filip Korzeniowski, David R.W. Sears, and Gerhard Widmer. A large-scale study of language models for chord prediction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.
- Florian Krebs, Sebastian Böck, Matthias Dorfer, and Gerhard Widmer. Downbeat tracking using beat synchronous features with recurrent neural networks. In *17th International Society for Music Information Retrieval Conference, (ISMIR)*, 2016.

- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- Mika Kuuskankare. ENP: A system for contemporary music notation. *Contemporary Music Review*, 28(2):221–235, 2009.
- Jonas Langhabel, Robert Lieck, Marc Toussaint, and Martin Rohrmeier. Feature discovery for sequential prediction of monophonic music. *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- Stefan Lattner, Maarten Grachten, and Gerhard Widmer. A predictive model for music based on learned interval representations. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.
- Bernhard Lehner, Gerhard Widmer, and Sebastian Bock. A low-latency, real-time-capable singing voice detection method with LSTM recurrent neural networks. In *23rd European signal processing conference (EUSIPCO)*, 2015.
- Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.
- Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22:55, 1932.
- Corentin Louboutin and Frédéric Bimbot. Description of chord progressions by minimal transport graphs using the system & contrast model. In *Proceedings of the 42nd International Computer Music Conference (ICMC)*, 2016.
- Bruce Lowerre. *The HARPY speech recognition system*. PhD thesis, Carnegie Mellon University, 1976.
- Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.
- Andrew McLeod and Mark Steedman. Evaluating automatic polyphonic music transcription. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

- Gabriele Medeot, Srikanth Cherla, Katerina Kosta, Matt McVicar, Samer Abdallah, Marco Selvi, Ed Newton-Rex, and Kevin Webster. StructureNet: Inducing structure in generated melodies. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *11th Annual Conference of the International Speech Communication Association (INTER-SPEECH)*, 2010.
- Saumitra Mishra, Bob L. Sturm, and Simon Dixon. Local interpretable model-agnostic explanations for music content analysis. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014.
- Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129):2, 1978.
- Emilio Molina, Ana M. Barbancho, Lorenzo J. Tardón, and Isabel Barbancho. Evaluation framework for automatic singing transcription. *15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.
- Marcel Mongeau and David Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24(3):161–175, 1990.
- Robert A. Moog. MIDI: musical instrument digital interface. *Journal of the Audio Engineering Society*, 34(5):394–404, 1986.
- Daniel Müllensiefen, Bruno Gingras, Lauren Stewart, and J Musil. The Goldsmiths Musical Sophistication Index (Gold-MSI): Technical report and documentation v1.0. *London: Goldsmiths, University of London*, 2011.
- Daniel Müllensiefen, Bruno Gingras, Jason Musil, and Lauren Stewart. The musicality of non-musicians: an index for assessing musical sophistication in the general population. *PloS one*, 9(2):e89642, 2014.
- Meinard Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.
- Kevin P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

- Eita Nakamura, Masatoshi Hamanaka, Keiji Hirata, and Kazuyoshi Yoshii. Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.
- Eita Nakamura, Kazuyoshi Yoshii, and Haruhiro Katayose. Performance error detection and post-processing for fast and accurate symbolic music alignment. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- Eita Nakamura, Emmanouil Benetos, Kazuyoshi Yoshii, and Simon Dixon. Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.
- Han-Wen Nienhuys and Jan Nieuwenhuizen. LilyPond, a system for automated music engraving. In *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, 2003.
- Yuta Ojima, Katsutoshi Itoyama, and Kazuyoshi Yoshii. A hierarchical Bayesian model of chords, pitches, and spectrograms for multipitch analysis. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, pages 1–13, 2018.
- Hélène Papadopoulou and Geoffroy Peeters. Joint estimation of chords and downbeats from an audio signal. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 19(1):138–152, 2010.
- Hyunsin Park and Chang D. Yoo. Melody extraction and detection through LSTM-RNN with harmonic sum loss. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.
- Joan Pastor-Pellicer, Francisco Zamora-Martínez, Salvador España-Boquera, and María José Castro-Bleda. F-measure as the error function to train neural networks. In *International Work-Conference on Artificial Neural Networks*, 2013.
- Christine Payne. Musenet. <https://openai.com/blog/musenet>, 2019.
- Marcus T. Pearce. Statistical learning and probabilistic prediction in music cognition: mechanisms of stylistic enculturation. *Annals of the New York Academy of Sciences*, 2018.

- Marcus T. Pearce and Geraint A. Wiggins. Expectation in melody: The influence of context and learning. *Music Perception: An Interdisciplinary Journal*, 23(5):377–405, 2006.
- Graham E. Poliner and Daniel P. W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 5(1):154–162, Oct 2006.
- Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *14th international workshop on content-based multimedia indexing (CBMI)*. IEEE, 2016.
- Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M. Schmidt, Andreas F. Ehmman, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *19th International Society for Music Information Retrieval Conference, (ISMIR)*, 2018.
- Lawrence Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.
- Stanislaw A. Raczynski, Emmanuel Vincent, and Shigeki Sagayama. Dynamic Bayesian networks for symbolic polyphonic pitch modeling. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 21(9):1830 – 1840, 2013.
- Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. PhD thesis, Columbia University, 2016.
- Colin Raffel and Daniel P.W. Ellis. Extracting ground-truth information from MIDI files: A MIDIfesto. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Dan P. W. Ellis. mir\_eval: A transparent implementation of common MIR metrics. In *15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.
- Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *International Conference on Machine Learning (ICML)*, 2018.
- Martin Rohrmeier and Thore Graepel. Comparing feature-based models of harmony. In *Proceedings of the 9th International Symposium on Computer Music Modeling and Retrieval (CMMR)*, 2012.

- Perry Roland. The music encoding initiative (MEI). In *Proceedings of the First International Conference on Musical Applications Using XML*, 2002.
- Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. An end-to-end framework for audio-to-score music transcription on monophonic excerpts. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- Matti P. Ryyanen and Anssi Klapuri. Polyphonic music transcription using note event modeling. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2005.
- Rodrigo Schramm, Helena De Souza Nunes, and Cláudio Rosito Jung. Audiovisual tool for solfège assessment. *ACM Transactions on Multimedia Computing, Communications, and Applications.*, 13(1), 2016.
- David R.W. Sears, Marcus T. Pearce, William E. Caplin, and Stephen McAdams. Simulating melodic and harmonic expectations for tonal cadences using probabilistic models. *Journal of New Music Research*, 47(1):29–52, 2018.
- Burr Settles. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 1–18, 2011.
- Siddharth Sigtia, Emmanouil Benetos, Srikanth Cherla, Tillman Weyde, Artur d’Avila Garcez, and Simon Dixon. An RNN-based music language model for improving automatic music transcription. *15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.
- Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 24(5):927–939, 2016.
- Paris Smaragdis and Judith C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2003.
- Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado University at Boulder, Department of Computer Science, 1986.

- Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In *19th International Society for Music Information Retrieval Conference, (ISMIR)*, 2018.
- Bob Sturm, João Felipe Santos, Oded Ben-Tal, and Iryna Korshunova. Music transcription modelling and composition using deep learning. In *1st Conference on Computer Simulation of Musical Creativity (CSMC)*, 2016.
- Li Su and Yi-Hsuan Yang. Combining spectral and temporal representations for multipitch estimation of polyphonic music. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(10):1600–1612, 2015.
- Y. Cem Subakan and Paris Smaragdis. Diagonal RNNs in symbolic music modeling. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017.
- Zheng Sun, Jiaqi Liu, Zewang Zhang, Jingwen Chen, Zhao Huo, Ching Hua Lee, and Xiao Zhang. Composing music with grammar argued neural networks and note-level encoding. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2018.
- Ilya Sutskever, Geoffrey E. Hinton, and Graham W. Taylor. The recurrent temporal restricted Boltzmann machine. In *Advances in Neural Information Processing Systems*, 2009.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.
- David Temperley. A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research*, 38(1):3–18, 2009.
- John Thickstun, Zaid Harchaoui, and Sham Kakade. Learning features of music from scratch. *International Conference on Learning Representations (ICLR)*, 2017.
- Karen Ullrich and Eelco van der Wel. Music transcription with convolutional sequence-to-sequence models. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop*. ISCA, 2016.

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
- Valerio Velardo, Mauro Vallati, and Steven Jan. Symbolic melodic similarity: State of the art and future challenges. *Computer Music Journal*, 40(2):70–83, 2016.
- Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.
- Christian Walder. Modelling symbolic music: Beyond the piano roll. In *Asian Conference on Machine Learning*, 2016.
- Christian Walder and Dongwoo Kim. Neural dynamic programming for musical self similarity. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.
- Chris Walshaw. The ABC music standard 2.1. <http://abcnotation.com/wiki/abc:standard:v2>, 1, 2011.
- Qi Wang, Ruohua Zhou, and Yonghong Yan. Polyphonic piano transcription with a note-based music language model. *Applied Sciences*, 8(3), 2018.
- Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
- Chih-Wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, Meinard Muller, and Alexander Lerch. A review of automatic drum transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(9):1457–1483, 2018.
- Yu Zhang and Qiang Yang. An overview of multi-task learning. *National Science Review*, 5(1):30–43, 09 2017.
- Umut Şimşekli, Jonathan Le Roux, and John R. Hershey. Hierarchical and coupled non-negative dynamical systems with application to audio modeling. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2013.