Learning and Evaluation Methodologies for Polyphonic Music Sequence Prediction with LSTMs

Adrien Ycart, *Student Member, IEEE,* Emmanouil Benetos, *Member, IEEE*

***Abstract*—Music language models (MLMs) play an important role for various music signal and symbolic music processing tasks, such as music generation, symbolic music classification, or automatic music transcription (AMT). In this paper, we investigate Long Short-Term Memory (LSTM) networks for polyphonic music prediction, in the form of binary piano rolls. A preliminary experiment, assessing the influence of the timestep of piano rolls on system performance, highlights the need for more musical evaluation metrics. We introduce a range of metrics, focusing on temporal and harmonic aspects. We propose to combine them into a parametrisable loss to train our network. We then conduct a range of experiments with this new loss, both for polyphonic music prediction (intrinsic evaluation) and using our predictive model as a language model for AMT (extrinsic evaluation). Intrinsic evaluation shows that tuning the behaviour of a model is possible by adjusting loss parameters, with consistent results across timesteps. Extrinsic evaluation shows consistent behaviour across timesteps in terms of precision and recall with respect to the loss parameters, leading to an improvement in AMT performance without changing the complexity of the model. In particular, we show that intrinsic performance (in terms of cross entropy) is not related to extrinsic performance, highlighting the importance of using custom training losses for each specific application. Our model also compares favourably with previously proposed MLMs.**

***Index Terms*—Music Language Models, Polyphonic Music Sequence Prediction, Automatic Music Transcription, Recurrent Neural Networks, Long Short-Term Memory.**

I. Introduction

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 in natural language and music theory. 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, such as dialect or musical style.

Such computational models of word sequence likelihoods, 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), that we define as computational models of music sequence likelihoods, 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 or symbolic music classification, as they could capture some stylistic aspects of music, and music cognition, by modelling music expectation.

One of the most notable uses of natural language models is speech recognition, where they have been combined with *acoustic models* for a long time, and greatly contribute to the success of today's methods [1]. The musical equivalent of speech recognition is automatic music transcription (AMT). Roughly, AMT is the task of extracting from a music audio signal a symbolic representation describing what notes were played, and when, usually in the form of a time-pitch matrix called *piano roll*, or a list of note events characterised by their pitch, onset time and duration [2]. For AMT, MLMs have only been introduced fairly recently, and are not explicitly used in most state-of-the-art systems [3], [4].

Western art 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 [5]. We focus here on polyphonic music sequence modelling, 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 [6]. In particular, long short-term memory (LSTM) networks [7] have helped make tremendous progress in natural language modelling [8]. In this paper, we propose to study LSTMs for polyphonic music sequence prediction, focusing on Western art music. Instead of building increasingly sophisticated architectures hoping to obtain better results, we propose to investigate com-

A. Ycart and E. Benetos are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, UK. e-mail: {a.ycart, emmanouil.benetos}@qmul.ac.uk. EB is supported by RAEng Research Fellowship RF/128.

prehensively 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. We also propose new training losses to adjust the behaviour of the model accordingly.

Although we use the term MLM to refer to our LSTM model, such a system should not be mistaken for a comprehensive model 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 more of a logical nature than statistical regularities. A simple LSTM network, despite showing some success in sequence modelling tasks, 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 the semantic content of music, we believe that this can still constitute useful biases for extrinsic tasks such as AMT. This study aims at investigating what kinds of patterns an LSTM is able to model, and to what extent it succeeds.

This study is the continuation of a previous pilot study on the same topic [9]. In the previous study, we compared the influence of various parameters, the most important being the timestep used for the analysis. We highlighted the fact that using a 10ms timestep yields very good prediction performance, but results in a poor model of musically-relevant sequential structure, only performing a simple temporal smoothing. On the other hand, sixteenth-note timesteps have a lower prediction performance, but exhibit more interesting musical properties, such as predicting when note transitions might happen and which notes are likely. We stated that new evaluation metrics were needed, as this difference in behaviour was not reflected by traditional metrics. In the present study, our main contributions are to:

- compare more timestep configurations, both quantitatively and qualitatively;
- propose new evaluation metrics that allow to highlight musically-relevant features of MLMs;
- design a parametric training loss based on the new metrics;
- investigate the relation between loss parameters, MLM performance and AMT performance.
- compare our model against two models found in the literature [10], [11].

The remainder of this paper is organised as follows. In Section II, we review existing works on music language modelling and their application to AMT. In Section III, we present the experimental setup we will use throughout our experiments. In Section IV, we present the results of a preliminary experiment comparing various timestep configurations with benchmark metrics. In Section V, 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 VI, 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 VII, we discuss the accomplished work and propose new directions for developing neural network-based prediction models.

II. Related works

A. Monophonic vs. polyphonic MLMs

Models of monophonic music sequences can be relatively simple. Markov models, such as n-grams and hidden Markov models (HMMs) can be used to model statistical regularities in monophonic music sequences with reasonable success, using for instance one state per possible note [12]. Although it is outside the scope of this paper to make a comprehensive review of such models, some notable examples can be cited. Some models of monophonic music were designed based on The Generative Theory of Tonal Music (GTTM) [13], 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 [14]; some recent work attempted to adapt it to polyphonic music [15]. Taking a rather different approach, IDyOM [16] is based on varying-order Markov chains, using a multiple-viewpoint framework. It was shown to correlate with human expectation of music. Recent work applied it to homophonic representations of polyphonic music [17].

However, these approaches can not be trivially adapted for polyphonic music. Indeed, for monophonic music, the number of distinct 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 a too high number to be computationally tractable. 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 large to be explored.

B. Neural MLMs

In the last few years, neural networks have been applied for polyphonic music sequence modelling. It has to be noted that polyphonic music sequence prediction has been used as an evaluation task in a variety of studies focusing on LSTMs [18]–[20], although their main goal is not music modelling. These studies use music sequence prediction as a benchmark task to compare LSTM variants and hyperparameter configurations, not as a goal in itself; they do not look into how the LSTM works depending on the input data. Some other networks were designed more specifically for music. In [10], a neural network architecture combining an RNN with a Restricted Boltzman Machine (RBM) was proposed to estimate at each timestep a pitch distribution, given the previous pitches. An LSTM for symbolic music modelling was also proposed in [21], with an original representation using a variable timestep, where the time difference between two timesteps is an extra variable predicted by the model. In [11], a variant of the LSTM unit using diagonal recurrent matrices was proposed and applied to polyphonic music sequence modelling. A neural architecture that can be trained to explicitly enforce transposition invariance properties was also proposed in [22]. It is used for monophonic music prediction, but can trivially be applied to polyphonic music by replacing the softmax outputs with sigmoids.

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 papers, such as [23], use music self-similarity to make predictions, based on the idea that human music is very likely to repeat itself with some slight variations. We focus here on the former approach.

C. MLMs for AMT

It is only fairly recently that some computational models of polyphonic music have been designed specifically for audio analysis. Aside from early attempts to include some musicological priors to transcription systems [24], Temperley [25] was one of the first to propose a joint probabilistic model for harmony, rhythm and stream separation, and suggested its use for audio transcription. This model is quite comprehensive; however, it seems to be intractable in real-life applications and has not been successfully applied to AMT. Since then, various AMT systems have used probabilistic models of high-level musical knowledge in order to improve their performance [26], [27], we choose not to review them in detail here.

More recently, the RNN-RBM model of [10] was integrated in various AMT systems. In [28], the same model was used as a transduction system to post-process the output of a Deep Belief Network front-end acting as an acoustic model. In [29], the RNN-RBM was used to refine the output of a Principal Latent Component Analysis (PLCA) multi-pitch estimator, using an iterative workflow. It was also integrated in [30] in an end-to-end neural-network, coupled with a variety of neural network-based acoustic models. Another AMT system was proposed in [31] using the same model, but in an original fashion. Similarly to [21], this one operates on frames that can have arbitrary durations, corresponding to inter-onset intervals. Despite being named "note-based", it does not process sequences of note events characterised by their pitch, onset time and duration. Its data format is still sequences of multiple-hot vectors, like the usual frame-based models, the difference being that the duration of the frames is not constant. In [32], the authors proposed to combine a simpler, LSTM-based MLM with a neural acoustic model; a blending model is used to dynamically weight the predictions made by the acoustic and language models.

D. Limitations

One key point, that appears to be often overlooked in the literature with frame-based models is the choice of a relevant timestep. Indeed, in [10], when the RNN-RBM was first introduced, it was used with a timestep corresponding to a fraction of a beat (sixteenth or eighth note). However, in [28]–[30], it was used with a timestep of the order of 10ms, which is both very short 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 [30], but not further investigated). Similar ideas were explored in [33], 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. They suggest though that on the chord-level (*i.e.* one symbol per chord, no matter how long), RNNs significantly outperform HMMs. In [9], we compared 10ms timesteps against 16th note tempo-related timesteps for polyphonic music sequence modelling. We suggested that with 10ms timesteps, a neural network would do little more than some kind of temporal smoothing, while with 16th note timesteps, it was able to model some basic music sequence patterns, such as periodicity in note transitions and the pitch distribution of a piece. In [32], we showed that this observation translated into better performance when combined with an acoustic model: using a language model operating on 16th note frames (using ground-truth beat positions) yielded better results than with 40ms frames. However, in terms of symbolic prediction, this observation was only qualitative, in the absence of relevant evaluation metrics. Moreover, we did not disentangle the influence of the length of the timestep vs. whether it is tempo-normalised. We will start by further investigating these questions in a preliminary experiment, whose results are presented in Section IV.

Additionally, despite all the work done towards using high-level musical knowledge in AMT systems, many systems achieve good performance without explicitly using MLMs. By contrast, in speech recognition, most system include a language model, and even simple ones such as trigrams can halve the word error rate of a system [34]. In [3], it was shown that [30] can be outperformed on AMT with neural acoustic models without using an MLM by carefully tuning hyperparameters and using appropriate input representations (although the authors did not investigate whether using an MLM could further improve results). In [31], 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. The current state of the art in AMT is [4], a neural model estimating separately the onsets and pitches of notes, 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. Against this evidence, our goal is to study the performance and shortcomings of MLMs. Although sophisticated architectures might improve MLM performance, they also complicate the systematic analysis of how model performance varies with different input, parameters and evaluation measures. Therefore, we use a simple LSTM model, we assess its performance depending on various parameters such as the input representation and the loss used for training, and compare it to other models.

III. Experimental setup

A. 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 natural language processing [35]. 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 $T \times 88$ matrix M, where T is the number of timesteps, and 88 corresponds to the number of keys on a piano, between MIDI notes A0 and C8. M is binary, such that $M_{t,p} = 1$ if and only if pitch p is active at timestep t. In particular, held notes and repeated notes are not differentiated. The output 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-vector corresponding to the t-th timestep of M, for all $t \in [\![0, T-1]\!]$, we try to predict M_{t+1} given the ordered sequence of $\{M_i\}_{i\in[\![0,t]\!]}$. We call the non-binary predicted matrix $\hat{M}$, and the binarised predicted matrix $\tilde{M}$. We index $\hat{M}$ and $\tilde{M}$ by $t \in [\![1, T]\!]$, 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 $\mathbb{t}$ corresponding to the times in seconds of the timesteps (*e.g.* $\mathbb{t} = [0, 0.01, 0.02...]$ 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 $N = (p_n, s_n, e_n)_n$ the sequence of notes in a given piece, where 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:

$$M_{t,p} = 1 \Leftrightarrow \exists n | p_n = p \wedge (s_n \leq \mathbb{t}_t + T_s) \wedge (e_n > \mathbb{t}_t + T_e) \quad (1)$$

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-timesteps piano roll, and choose the values that maximise the framewise F-Measure over the whole dataset between these two piano rolls through grid search.

B. 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[1].

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. 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 V-A), 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 IV), 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 III-C), and we use one single threshold for all pitches.

C. Dataset

1) The Piano-midi.de dataset: We use the Piano-midi.de dataset[2] as real-world MIDI data. 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 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

[1]https://github.com/adrienycart/PolyMusicPredLSTM
[2]http://piano-midi.de/

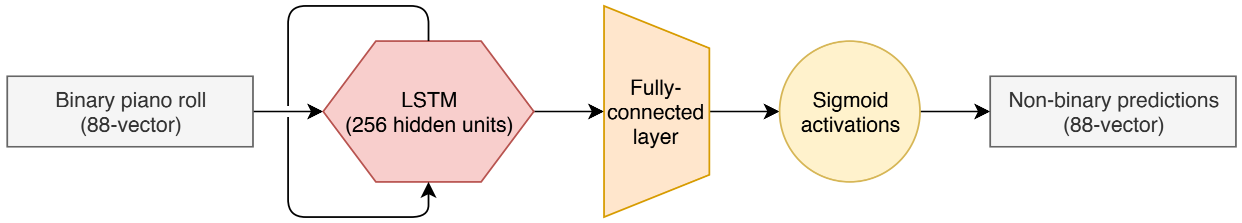

Fig. 1. Single-layer LSTM network architecture. The non-binary predictions can then be thresholded to obtain binary predictions.

using this dataset is that the MAPS dataset [36], 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[3].

2) 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 [10]. 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.

IV. Preliminary experiment

A. Description

As a first experiment, we reproduce and extend the results of [9]. 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 III-B.
- *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 V-A, 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 [33], the difference being that there might be repeated frames when the same note 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.

B. Quantitative analysis

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

[3]http://c4dm.eecs.qmul.ac.uk/ycart/taslp20.html

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

TABLE I
PREDICTION PERFORMANCE FOR VARIOUS TIMESTEPS ASSESSED WITH THE BENCHMARK METRICS (DEFINED IN SECTION V-A). BOLD VALUES CORRESPOND TO THE BEST RESULT FOR EACH TIMESTEP.

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 [33].

C. 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 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*), 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 timestep. 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.

V. EVALUATION METRICS

A. 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 [37], namely the frame-wise precision, recall and F-Measure. These measures are defined as follows:

$$\mathcal{P} = \frac{\sum_{t=0}^{T} TP(t)}{\sum_{t=0}^{T} TP(t) + FP(t)} \quad (2)$$

$$\mathcal{R} = \frac{\sum_{t=0}^{T} TP(t)}{\sum_{t=0}^{T} TP(t) + FN(t)} \quad (3) \qquad \mathcal{F} = \frac{2 \cdot P \cdot R}{P + R} \quad (4)$$

where $TP(t)$, $FP(t)$ and $FN(t)$ are the number of true positives, false positives and false negatives, respectively, when comparing $\tilde{M}$ and M at timestep t. We count one true positive for each (t, p) such that $\tilde{M}_{t,p} = M_{t,p} = 1$.

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:

$$\mathcal{H}(M_t, \hat{M}_t) = -\sum_{p \in [\![0,87]\!]} M_{t,p} \log(\hat{M}_{t,p}) + (1 - M_{t,p}) \log(1 - \hat{M}_{t,p}) \quad (5)$$

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.

B. 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.

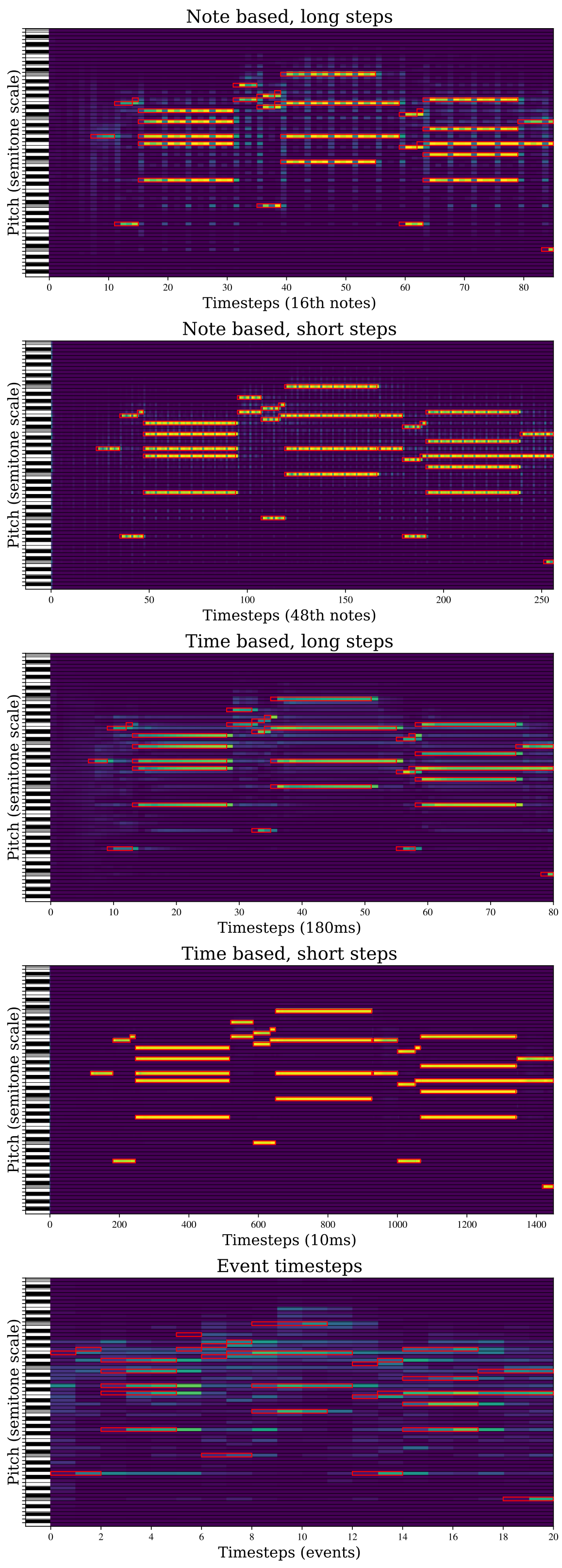

Fig. 2. Preliminary experiment: Comparison of sigmoid outputs for various timesteps for the MIDI file `chpn-p7.mid` transposed into C. Ground truth notes are overlaid as red rectangles. The notes of the scale correspond to the white keys on the left of each image. The tonic is in grey. The same color map is used across figures.

1) *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 $[\![1, T-1]\!]$ such that:

$$t \in Tr \Leftrightarrow M_t \neq M_{t-1} \tag{6}$$

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:

$$\mathcal{H}_{tr} = \frac{1}{|Tr|} \sum_{t \in Tr} \frac{\mathcal{H}(M_t, \hat{M_t})}{d(t)} \tag{7}$$

where $|.|$ denotes the cardinality. We divide by $d(t)$ as we observe experimentally that $H(M_t, \hat{M_t})$ 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.

2) *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:

$$\mathcal{H}_{ss} = \frac{1}{T-1-|Tr|} \sum_{t \in [\![1,T-1]\!] \setminus Tr} \mathcal{H}(M_t, \hat{M_t}) \tag{8}$$

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.

3) *Pitch-profile Cross entropy:* In Section IV, we observed that the network seems to be able to recognise the pitch distribution in certain cases, and 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 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 V-C2. 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 $K_i = [\![t_i, t_{i+1}[\![$. Let $N_{p,i}$ the subset of N such that:

$$N_{p,i} = \{(p_n, s_n, e_n) \in N | p_n = p \wedge (s_n \in K_i \vee e_n \in K_i)\} \tag{9}$$

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

$$p \in \mathrm{P}(t) \iff \frac{\sum_{N_{p,i}} \min(e_n, t_{i+1}) - \max(s_n, t_i)}{t_{i+1} - t_i} > 0.05 \tag{10}$$

We subsequently define a scale vector $P(t)$ such that $P(t)_p = 1 \iff p \in \mathrm{P}(t)$. $P(t)$ 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:

$$B = \{(t, p) \in [\![1, T-1]\!] \times [\![0, 87]\!] \mid M_{t,p} = 0\} \tag{11}$$

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:

$$\mathcal{H}_{pp}(t, p) = -P(t)_p \log(\hat{M}_{t,p}) - (1 - P(t)_p) \log(1 - \hat{M}_{t,p}) \tag{12}$$

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:

$$\mathcal{H}_{pp} = \frac{1}{|B|} \sum_{(t,p) \in B} \mathcal{H}_{pp}(t, p) \tag{13}$$

4) 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 $B_t = \{p \in [\![0, 87]\!] \mid (t, p) \in B\}$. We then have:

$$\mathcal{H}_{pp,tr} = \frac{1}{\sum_{t \in Tr} |B_t|} \sum_{t \in Tr, p \in B_t} \mathcal{H}_{pp}(t, p) \tag{14}$$

5) Steady-State-Pitch-profile Cross entropy: By analogy with $\mathcal{H}_{ss}$(see Section V-B2), we also define the steady-state-pitch-profile cross entropy:

$$\mathcal{H}_{pp,ss} = \frac{1}{\sum_{t \notin Tr} |B_t|} \sum_{t \notin Tr, p \in B_t} \mathcal{H}_{pp}(t, p) \tag{15}$$

C. 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.

1) $\mathcal{H}_{tr}$ 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 that scores both good $\mathcal{H}_{tr}$ and good $\mathcal{H}_{ss}$ is thus a great model, both able to know when a note will be held, and to successfully predict 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* timesteps. 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.

2) $\mathcal{H}_{pp}$, $\mathcal{H}_{pp,tr}$ 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.

D. 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 $\Theta = (w_{tr}, w_{ss}, \alpha)$:

$$\mathcal{S}_\Theta = \sqrt{(w_{tr}\mathcal{H}_{tr} + w_{ss}\mathcal{H}_{ss})^{1+\alpha}(w_{tr}\mathcal{H}_{pp,tr} + w_{ss}\mathcal{H}_{pp,ss})^{1-\alpha}} \tag{16}$$

where $(w_{tr}, w_{ss}) \in \mathbf{R}_+^2$ and $\alpha \in [-1, 1]$. When $\Theta = (1, 1, 0)$, we omit the Θ 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 α 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 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 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.

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

A. 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 II. We also include values for $\mathcal{S}$ for completeness.

1) 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 IV, that the longer 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 IV-C 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 enough temporal context to sometimes predict with relative success note changes.

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

TABLE II
Prediction performance for various timesteps assessed with the proposed metrics. Bold values correspond to the best result for each timestep.

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.

2) 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 V-C2, 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.

3) 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.

4) 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.

B. 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 [10] and the diagonal RNN [11] (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 [11], 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 III, and we show some example outputs with *note-long* timesteps in Figure 3. Outputs with *note-short* timesteps can be found on our supplementary materials webpage[4].

In terms of $\mathcal{H}$, the best performing model is the LSTM trained with $\mathcal{H}$ loss. This conflicts with the conclusions from [11], 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 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.

[4]http://c4dm.eecs.qmul.ac.uk/ycart/taslp20.html

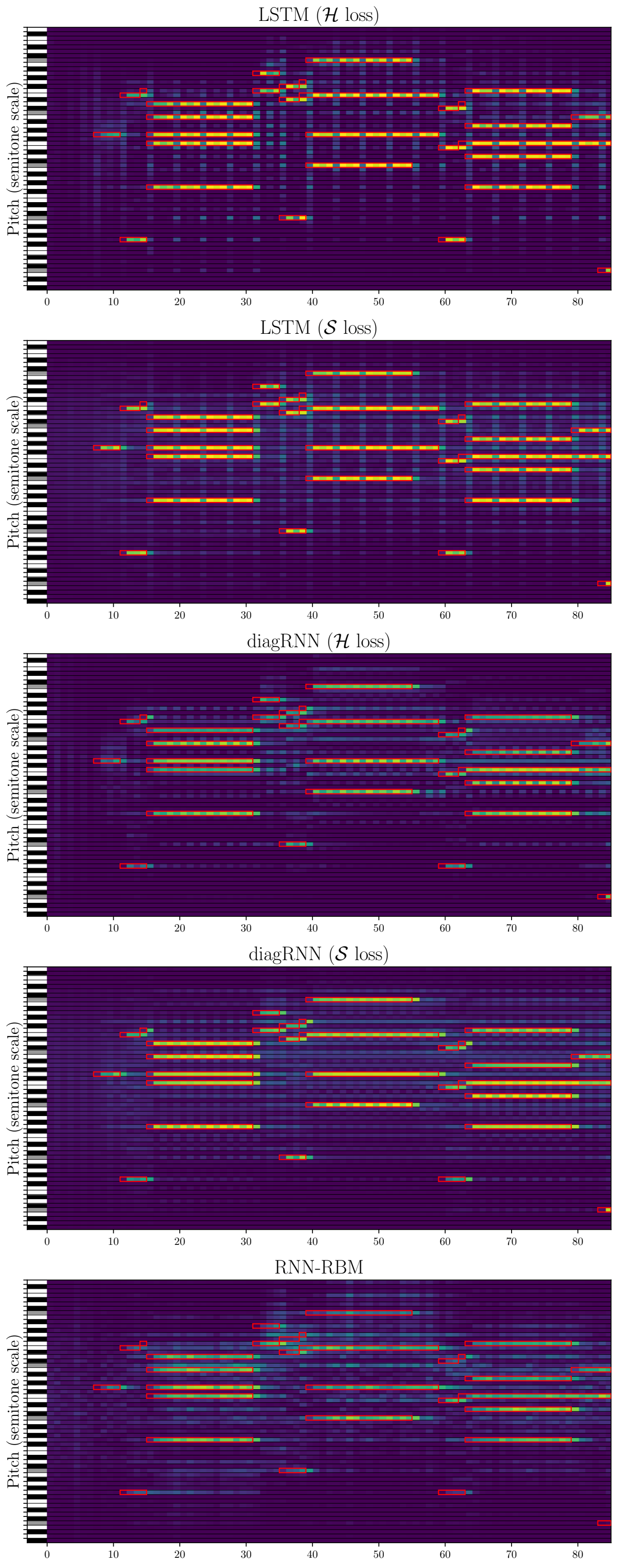

Fig. 3. Comparison of sigmoid outputs for various models for the MIDI file `chpn-p7.mid` transposed into C, with *note-long* timesteps. The ground truth notes are overlaid as red rectangles. The notes of the scale correspond to the white keys on the left of each image. The tonic is in grey. The same color map is used across images.

Model	$\mathcal{F}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{H}$	$\mathcal{H}_{tr}$	$\mathcal{H}_{ss}$	$\mathcal{H}_{pp}$	$\mathcal{H}_{pp,tr}$	$\mathcal{H}_{pp,ss}$	$\mathcal{S}$
NOTE-SHORT										
LSTM ($\mathcal{H}$ loss)	84.1	86.3	82.0	**2.76**	3.05	**0.524**	1.07	0.811	1.15	2.61
LSTM ($\mathcal{S}$ loss)	**84.7**	**89.0**	81.0	2.79	**2.81**	0.975	0.691	**0.548**	0.737	**2.16**
diagRNN [11] ($\mathcal{H}$ loss)	83.9	84.5	83.3	3.03	3.04	0.79	1.0	0.918	1.03	2.68
diagRNN [11] ($\mathcal{S}$ loss)	83.7	83.7	**83.7**	3.7	3.08	1.69	**0.601**	0.565	**0.613**	2.33
RNN-RBM [10]	83.1	83.4	82.9	3.49	3.3	1.12	0.889	0.85	0.902	2.74
NOTE-LONG										
LSTM ($\mathcal{H}$ loss)	**63.3**	64.8	**62.3**	**5.4**	2.46	1.83	0.856	0.736	0.995	2.65
LSTM ($\mathcal{S}$ loss)	60.6	**67.7**	55.8	6.19	2.81	**1.52**	0.59	0.532	0.667	**2.22**
diagRNN [11] ($\mathcal{H}$ loss)	61.2	61.3	61.4	5.42	**2.37**	2.04	0.775	0.735	0.843	2.6
diagRNN [11] ($\mathcal{S}$ loss)	58.6	58.6	58.6	6.66	2.91	1.59	**0.555**	**0.525**	**0.612**	2.23
RNN-RBM [10]	56.4	56.8	56.2	6.97	2.79	3.57	0.67	0.658	0.69	2.88

TABLE III
COMPARISON OF VARIOUS MLMS, FOR NOTE-SHORT AND NOTE-LONG TIMESTEPS.

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.

C. *Influence of* Θ *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 Θ 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. In line with [9], 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 4.

Inspecting the results of our MLMs with various Θ configurations, we can see that the parameters influence the end result as expected. When increasing the value of α, $\mathcal{H}$ 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 α improves $\mathcal{F}$. This makes sense as a higher α will make the network try to predict specific notes, while a lower α 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 Θ. 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 α 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 α 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 $\mathcal{H}$.

D. *Application to AMT*

1) 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 [32]. In other words, we post-process the output of the convolutional neural network (CNN) acoustic model described in [3]. At each timestep, we use our LSTM to make predictions for the next timestep based on the previous binary outputs of

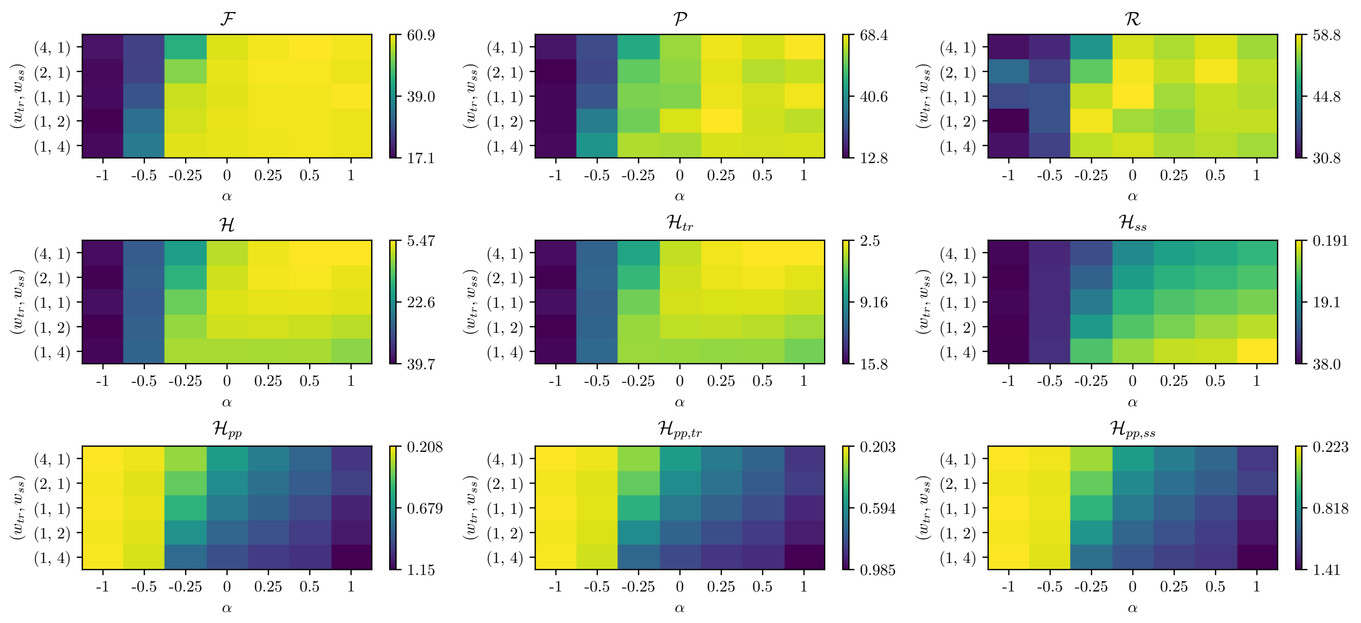
Fig. 4. Evaluation of various MLMs trained with $\mathcal{S}_\Theta$ loss, with various Θ configurations, all with *note-long* timesteps. Each grid corresponds to a metric defined in Section V, rows correspond to different (w_{tr}, w_{ss}) configurations, and columns correspond to different α. Brighter colours correspond to better performance (higher for $\mathcal{F}$, $\mathcal{P}$ and $\mathcal{R}$, lower for cross-entropy-based metrics).

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 [32]). Outputs for the next timestep are obtained by sampling from this combined distribution. The best sequence overall is then obtained with Viterbi decoding with beam search (see [32] for details).

As in [32], 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 [32], 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 [32], namely framewise and onset-only notewise $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{F}$. MLMs are trained using the setup described in the current paper, but use the same dataset splits as [32]. 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 Θ when using $\mathcal{S}_\Theta$ as MLM training loss influences AMT performance. Using the same grid search parameters as in Section VI-C, 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 show the full results of the grid search with 16th note timesteps in Fig. 5 (the results for other timesteps are available on our supplementary materials webpage[5]). AMT results for the baseline, cross-entropy-trained-MLM decoding, and best-performing MLM are shown in Table IV, along with the Θ parameters that yield best performance.

[5]http://c4dm.eecs.qmul.ac.uk/ycart/taslp20.html

2) Results: When considering the effect of Θ on AMT performance, we can observe similar tendencies for all tested timesteps. Using a lower α 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 α 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 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 Θ 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 [32], notewise $\mathcal{F}$ is below baseline for 16th note timesteps). This is a promising result, showing that by selecting appropriate Θ parameters according to the task to be addressed, an improvement can be obtained without increasing model complexity. Moreover, consistant behaviour in terms of $\mathcal{P}$ and $\mathcal{R}$ can be observed across timesteps depending on the tuning of the Θ 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

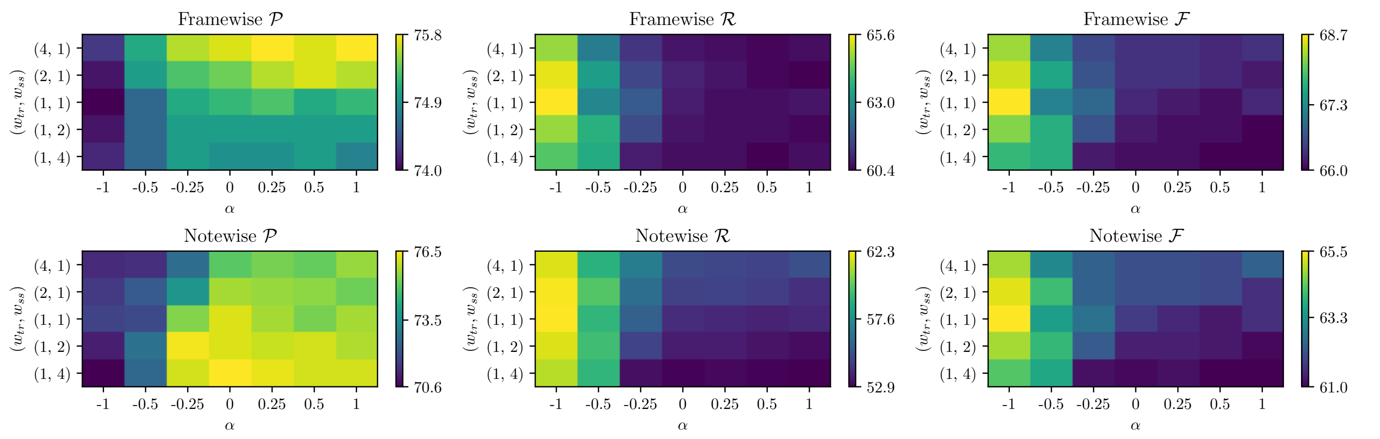

Fig. 5. AMT performance of various MLMs trained with the $\mathcal{S}_\Theta$ loss, with various Θ configurations. Each grid corresponds to a metric, rows correspond to different (w_{tr}, w_{ss}) configurations, columns correspond to different α. Brighter colours correspond to better values.

Model	Framewise			On-Notewise		
	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$
40ms timesteps						
Baseline [3]	72.6	**65.0**	**67.8**	51.1	**67.5**	56.8
LSTM ($\mathcal{H}$ loss)	**73.3**	64.1	67.6	60.6	64.1	61.2
LSTM ($\mathcal{S}_{(1,4,0.25)}$ loss)	73.0	64.4	67.7	**61.4**	64.0	**61.6**
16th note timesteps						
Baseline [3]	74.8	65.0	68.6	71.0	**63.8**	**66.1**
LSTM ($\mathcal{H}$ loss)	**76.1**	60.6	66.5	**75.7**	55.4	62.6
LSTM ($\mathcal{S}_{(1,1,-1)}$ loss)	74.0	**65.6**	**68.7**	71.8	62.3	65.5

TABLE IV
Comparison of AMT performance for various MLMs. All results are obtained using the CW configuration, as described in [32]. The best values for each timestep are in bold.

evaluate MLMs is thus not a good reflection of their ability to perform well in extrinsic tasks, at least for AMT.

VII. 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 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 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 [38].

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 [39]. 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 of 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, the current loss could not be trivially adapted. Still, investigating different kinds of representations is an important question in exploring polyphonic music sequence prediction.

Acknowledgments

The authors would like to thank Andrew McLeod, Marcus Pearce, Peter Harrison and Roger Dean for their useful suggestions on this work.

References

[1] X. Huang and L. Deng, "An overview of modern speech recognition," in *Handbook of Natural Language Processing*, 2010.

[2] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[3] R. Kelz, M. Dorfer, F. Korzeniowski, S. Bock, A. Arzt, and G. Widmer, "On the Potential of Simple Framewise Approaches to Piano Transcription," *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pp. 475–481, 2016.

[4] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[5] Z. Chen, J. Droppo, J. Li, W. Xiong, Z. Chen, J. Droppo, J. Li, and W. Xiong, "Progressive joint modeling in unsupervised single-channel overlapped speech recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 1, pp. 184–196, 2018.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Interspeech*, vol. 2, 2010, p. 3.

[9] A. Ycart and E. Benetos, "A study on LSTM networks for polyphonic music sequence modelling," in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 421–427.

[10] N. Boulanger-Lewandowski, P. Vincent, and Y. Bengio, "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription," *29th International Conference on Machine Learning*, pp. 1159–1166, 2012.

[11] Y. C. Subakan and P. Smaragdis, "Diagonal RNNs in symbolic music modeling," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2017 IEEE Workshop on*. IEEE, 2017, pp. 354–358.

[12] J. E. Cohen, "Information theory and music," *Behavioral Science*, vol. 7, no. 2, pp. 137–163, 1962.

[13] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. MIT Press, 1983.

[14] M. Hamanaka, K. Hirata, and S. Tojo, "Implementing "A generative theory of tonal music"," *Journal of New Music Research*, vol. 35, no. 4, pp. 249–277, 2006.

[15] K. H. Masatoshi Hamanaka and S. Tojo, "Polyphonic Music Analysis Database Based on GTTM," in *2nd Conference on Computer Simulation of Musical Creativity*, 2017.

[16] M. T. Pearce, "Statistical learning and probabilistic prediction in music cognition: mechanisms of stylistic enculturation," *Annals of the New York Academy of Sciences*, 2018.

[17] D. R. Sears, M. T. Pearce, W. E. Caplin, and S. McAdams, "Simulating melodic and harmonic expectations for tonal cadences using probabilistic models," *Journal of New Music Research*, vol. 47, no. 1, pp. 29–52, 2018.

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning*, 2014.

[19] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning*, 2015, pp. 2342–2350.

[20] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[21] C. Walder, "Modelling symbolic music: Beyond the piano roll," in *Asian Conference on Machine Learning*, 2016, pp. 174–189.

[22] S. Lattner, M. Grachten, and G. Widmer, "A Predictive Model for Music Based on Learned Interval Representations," in *19th International Society for Music Information Retrieval Conference*, 2018.

[23] C. Walder and D. Kim, "Neural dynamic programming for musical self similarity," in *Proceedings of the 35 th International Conference on Machine Learning*, 2018.

[24] M. P. Ryynanen and A. Klapuri, "Polyphonic music transcription using note event modeling," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* IEEE, 2005, pp. 319–322.

[25] D. Temperley, "A Unified Probabilistic Model for Polyphonic Music Analysis," *Journal of New Music Research*, vol. 38, no. 1, pp. 3–18, 2009.

[26] S. A. Raczyński, E. Vincent, and S. Sagayama, "Dynamic Bayesian networks for symbolic polyhonic pitch modeling," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 9, pp. 1830 – 1840, 2013.

[27] Y. Ojima, K. Itoyama, and K. Yoshii, "A Hierarchical Bayesian Model of Chords, Pitches, and Spectrograms for Multipitch Analysis," in *Proc. 17th International Society for Music Information Retrieval Conference*, 2016.

[28] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "High-dimensional Sequence Transduction," *Bernoulli*, no. 5, pp. 3178–3182, 2013.

[29] S. Sigtia, E. Benetos, S. Cherla, T. Weyde, A. d'Avila Garcez, and S. Dixon, "An RNN-based Music Language Model for Improving Automatic Music Transcription," *Proceedings of the 15th International Society for Music Information Retrieval Conference*, no. Ismir, pp. 53–58, 2014.

[30] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, May 2016.

[31] Q. Wang, R. Zhou, and Y. Yan, "Polyphonic piano transcription with a note-based music language model," *Applied Sciences*, vol. 8, no. 3, 2018.

[32] A. Ycart, A. McLeod, E. Benetos, and K. Yoshii, "Blending Acoustic and Language Model Predictions for Automatic Music Transcription," in *20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[33] F. Korzeniowski and G. Widmer, "On the Futility of Learning Complex Frame-Level Language Models for Chord Recognition," in *AES International Conference on Semantic Audio*, 2017.

[34] J. Chorowski and N. Jaitly, "Towards better decoding and language model integration in sequence to sequence models," in *Proc. Interspeech 2017*, 2017, pp. 523–527.

[35] D. Jurafsky, *Speech & Language Processing*. Pearson Education India, 2000.

[36] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 6, pp. 1643–1654, Aug. 2010.

[37] M. Bay, A. F. Ehmann, and J. S. Downie, "Evaluation of Multiple-F0 Estimation and Tracking Systems," in *10th International Society for Music Information Retrieval Conference (ISMIR)*, 2009, pp. 315–320.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[39] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

Adrien Ycart received the MSc. in signal processing from Télécom ParisTech and IRCAM, Paris. He is currently pursuing the Ph.D. degree at the Centre for Digital Music, Queen Mary University of London. He previously worked on rhythm transcription as a research engineer at IRCAM. His work focuses on symbolic music modelling with neural networks, and the use of such models as language models for automatic music transcription.

Emmanouil Benetos is currently Senior Lecturer at Queen Mary University of London and Turing Fellow at the Alan Turing Institute. He received his BSc and MSc degrees in informatics from Aristotle University of Thessaloniki, Greece, in 2005 and 2007, respectively, and his PhD degree in electronic engineering from Queen Mary University of London in 2012. In 2013-2015 he was University Research Fellow at City, University of London and in 2015-2020 he was Royal Academy of Engineering Research Fellow at Queen Mary University of London. He has published more than 100 peer-reviewed papers spanning several topics in audio and music signal processing. His research focuses on signal processing and machine learning for music and audio analysis as well as applications to music information retrieval, sound scene analysis, and computational musicology.