BLENDING ACOUSTIC AND LANGUAGE MODEL PREDICTIONS FOR AUTOMATIC MUSIC TRANSCRIPTION

Adrien Ycart[1*] **Andrew McLeod**[2*] **Emmanouil Benetos**[1] **Kazuyoshi Yoshii**[2]
[1] Queen Mary University of London, UK [2] Kyoto University, Japan
a.ycart@qmul.ac.uk, mcleod@sap.ist.i.kyoto-u.ac.jp

ABSTRACT

In this paper, we introduce a method for converting an input probabilistic piano roll (the output of a typical multi-pitch detection model) into a binary piano roll. The task is an important step for many automatic music transcription systems with the goal of converting an audio recording into some symbolic format. Our model has two components: an LSTM-based music language model (MLM) which can be trained on any MIDI data, not just that aligned with audio; and a blending model used to combine the probabilities of the MLM with those of the input probabilistic piano roll given by an acoustic multi-pitch detection model, which must be trained on (a comparably small amount of) aligned data. We use scheduled sampling to make the MLM robust to noisy sequences during testing. We analyze the performance of our model on the MAPS dataset using two different timesteps (40ms and 16th-note), comparing it against a strong baseline hidden Markov model with a training method not used before for the task to our knowledge. We report a statistically significant improvement over HMM decoding in terms of notewise F-measure with both timesteps, with 16th note timesteps improving further compared to 40ms timesteps.

1. INTRODUCTION

The ultimate goal of the task of Automatic Music Transcription (AMT) is to convert an audio signal into some form of human- or machine-readable music notation [2]. This process is divided into two main steps. First, an acoustic model performs multi-pitch detection by converting an input acoustic signal into a *posteriogram*: a pseudo-piano roll matrix which contains the probability of each pitch being present at each timestep according to the acoustic model. Next, a music language model (MLM) is used to enforce some musicality on the results, converting the posteriogram into a human-readable format.

While it is desirable to run these two models jointly—and some systems have been designed in such a way with success, either with relatively simple MLMs (e.g., [19,20]), or for performing a simpler task than AMT (e.g., chord detection [15])—the search space and resulting computation quickly becomes too large to be feasible for more complex probabilistic MLMs which explicitly model musical structure (e.g., [22]). Furthermore, such MLMs have typically been designed to take as input MIDI (or MIDI-like) data which consists of lists of musical notes, rather than the typical posteriogram output of acoustic models.

However, the conversion of a posteriogram into MIDI is not a trivial task, and has not been the focus of much research until recently. A naive approach is simple thresholding of the posteriogram, and a simple two-state (on/off) HMM has also been proposed [17]. Some more sophisticated models have attempted to use neural networks as implicit MLMs to incorporate some prior musical knowledge into their systems (e.g., [21,24]), but often, they bring only modest improvement [21] or the MLM is only used in rare occasions [24] (see Section 2 for a more complete discussion on these and other related systems). The main issues that we identify with these previous attempts to incorporate MLMs are (1) the MLM fails to capture musical features because of an inappropriately short timestep which over-emphasizes self-transitions [25], and (2) the MLM is not robust to noise during decoding.

Our main contributions are to:

1. compare the use of a musically-relevant timestep for MLM decoding—specifically a 16th note, as recommended in [25]—to the more standard 40ms.
2. train the MLM with scheduled sampling [3], making it more robust to noise at test time.
3. propose a novel "blending" model which dynamically merges probabilities from the acoustic model and the MLM rather than using a simpler method such as a linear combination.
4. describe a new training method for a previously-proposed post-processing HMM [17], leading to a significant improvement in F-measure over the standard maximum likelihood approach.

For contribution (1), note that in a realistic setting, using a 16th note timestep would require a beat-tracking algorithm. However, in this proof-of-concept experiment, we consider 16th note locations as given, and leave the integration of a noisy 16th note timestep for future work. It should be noted that a 16th note timestep was already investigated in [27] for polyphonic sequence transduction, which concluded that although an improvement is observed when using 16th note timesteps, it is attributed only to the fact

[*] Authors 1 and 2 contributed equally to this work.

that the resulting outputs are quantized to the ground truth metrical grid. By contrast, we show here that a 16th-note MLM brings improvement for language model decoding beyond quantisation of the output.

We conduct our experiments using a state-of-the-art piano-specific acoustic model [13] (although our system could be applied to a variety of musical instruments and styles). Overall, we show that our full system with the 16th note timestep, scheduled sampling, and the blending model, leads to a significant improvement in F-measure over both the baseline and the proposed HMM, with identical timesteps.

2. RELATED WORK

The most basic strategy for binarizing time-pitch posteriograms is simple thresholding, where outputs below some value are set to 0 while all others are set to 1 (e.g., in [9, 11, 13]). Slightly more sophisticated is to use a two-state (on/off) HMM for each pitch (proposed in [17], used in, e.g., [6, 7]), where results tend to be cleaner with fewer spurious notes using this method. Still, the musicality of such a model is limited, as it considers each pitch independently, and does not consider more than one previous frame for each transition.

Recently, deep learning methods have also been used for this task, typically using some form of RNN. They can be broadly grouped into performing one of two tasks: transduction or language model decoding.

Transduction methods aim to convert one sequence of symbols into another (here, the output of an acoustic model into a binary piano roll). Examples include [5], which uses an architecture combining an RNN with a Restricted Boltzmann Machine (RBM), and [27], which investigated the performance of an LSTM-based model. One drawback of transduction methods is that they require aligned MIDI and audio recordings for training. Similarly, they are trained on one specific acoustic model's outputs, and do not necessarily generalize to other acoustic models.

With *language model decoding* methods, an MLM is trained to assess the likelihood of an output sequence. Importantly, such a model is trained on symbolic data, independent of any acoustic model. For example, [21] uses an RNN-RBM as a language model, combined with various neural acoustic models. Similarly, [24] uses an RNN-RBM language model, but instead of using a fixed framerate, it operates on frames corresponding to detected inter-onset-intervals. Our method belongs to this second category of language model decoders, with the caveat that one component of it, the blending model, requires training on aligned pairs of input and output, though much less than would be needed to train a neural transduction model.

As mentioned, using an MLM has brought only limited improvement to the performance of AMT systems in the above studies. One reason for this lack of substantial improvement in [21] might be the use of an inappropriate timestep for language modelling: 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 analyzed. Indeed, [25] hints at the fact that for polyphonic music sequence prediction, using a small time-step only results in a smoothing effect due to the predominance of self-transitions, and using a musically-relevant time step such as a 16th note allows the network to learn more interesting musical properties. To that end, [24] describes an MLM which uses note-based timesteps. However, the MLM was only used in the rare case that a note onset was detected without a corresponding pitch. 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 [21]).

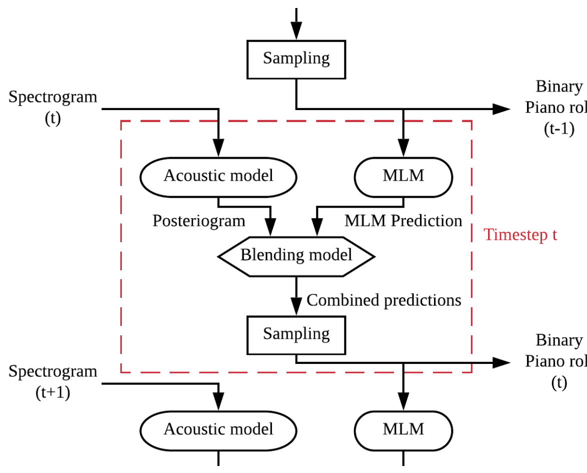

Figure 1. The proposed system.

3. PROPOSED SYSTEM

Our system takes as input a probabilistic piano roll, specifically the output of the acoustic model from [13]. That model is a CNN which takes as input a spectrogram with logarithmically-spaced frequency bins and log-magnitude with a timestep of 40ms, and is a benchmark acoustic model for piano transcription.

Our system's inputs are in the form of matrix $I \in \mathbb{R}^{N_p \times T}$, where T is the length of the input in frames, $N_p = 88$ (one row per key on a piano keyboard), and each element $I_{p,t}$ contains the probability of a pitch p being present at frame t. Our output is the binary matrix $O \in \{0,1\}^{N_p \times T}$, where $O_{p,t}$ is 1 if pitch p is present at frame t, and 0 otherwise.

Our system flow is shown in Fig. 1. It consists of two main components: an LSTM-based language model (see Section 3.1), which predicts the presence of each pitch at a frame given the previous frames; and a blending model (see Section 3.2), which combines the input 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 3.3.

3.1 Language Model

The language model has the same architecture as described in [25]. It is a single-layer LSTM, with a hidden layer of size 256 and 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. Two different MLMs were trained, one operating on 40ms timesteps, and one on 16th note timesteps. We use $L_{p,t}$ to denote the MLM's output corresponding to pitch p at frame t.

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 prediction 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, as noted in [21] and [24]. To solve this problem, we use scheduled sampling [3]: 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 predictions made by the MLM at the previous timestep. Training starts with $p_{GT} = 1$, and p_{GT} is decreased as training progresses, 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) [3] mentions 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.

3.2 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 job is to learn the situations in which each model performs well, and output the combined prior for a pitch at a frame based on both the probabilities from the acoustic model and the MLM, as well as some surrounding context.

It is a feed-forward neural network with l [1] hidden layers 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 ($I_{p,t}$ and $L_{p,t}$), (2) the sample history at that pitch for the previous $hist$ [1] frames ($O_{p,t'}$ for $max(0, t - hist) \leq t' < t$), and (3) nine additional hand-crafted features, described in Table 1, resulting in an input vector of length $11 + hist$.

The sample history allows the model to learn if there are certain situations in which the LSTM performs particularly well or poorly. Features 1–4 model how peaked the

Feature	Description	Equation
1–2	Uncertainty	Eqn. (1)
3–4	Entropy	Eqn. (2)
5–6	Mean	$\sum_{p'<N_p} \frac{I_{p',t}}{N_p}$
7–8	Flux	$I_{p,t} - I_{p,t-1}$
9	Pitch	$\frac{p}{N_p}$

Table 1. The features used for the blending model. For equations written using I, the second feature is calculated identically with L.

output distribution from each model is (and thus how certain it might be) 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.

$$\text{Uncertainty} = \sum_{p'<N_p} \begin{cases} (1 - I_{p',t})^2 & I_{p',t} > 0.5 \\ (I_{p',t})^2 & I_{p',t} \leq 0.5 \end{cases} \quad (1)$$

$$\text{Entropy} = \frac{1}{log_2(N_p)} \sum_{p'<N_p \wedge I_{p',t} \neq 0} -I_{p',t} log_2(I_{p',t}) \quad (2)$$

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 $P_{p,t}$ as a weighted sum: $P_{p,t} = w_{p,t} I_{p,t} + (1 - w_{p,t}) L_{p,t}$. Second, a prior model (PM) which outputs $P_{p,t}$ directly. The main difference between the two models is that WM can only ever result in a $P_{p,t}$ that lies somewhere between $I_{p,t}$ and $L_{p,t}$, while PM can always output any $P_{p,t}$ between 0 and 1.

3.3 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 [23] with beam search using a beam of size b and a branching factor k. Specifically, at each frame, we save only the b most probable histories to that point. Then, for each of those at frame t, using the blending model's output distribution $P_{p,t}$, we sample the k most probable samples using Algorithm 2 from [5] (again saving only the top b from the $b * k$ resulting hypotheses). 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 $P_{p,t}$ is:

$$P(S_t) = \prod_{p' \in S_t} P_{p',t} \prod_{p'<N_p \wedge p' \notin S_t} 1 - P_{p',t} \quad (3)$$

Beam search has the drawback that the beam can easily become saturated with only slight variations of the most probable hypothesis. Therefore, similar to [21] and [15], 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 O of our system is constructed using the sample history of the most probable state in the beam, such that $O_{p,t}$ is 1 if $p \in S_t$ and 0 otherwise.

[1] See Section 4.5 for details on the training of l and $hist$.

4. EXPERIMENTS

4.1 Data

For our experiments, we use the MAPS dataset [10], 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. We create the exact same test set as was used in [13] (which was created in the same way as *Configuration II* from [21], 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 1), and use the remaining pieces for training. This results in final split sizes of 59 pieces for test, 105 for training, and 32 for validation. The decrease in training set size compared to [13] 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 [13].

To train our MLM, we use MIDI files taken from the Piano-midi.de [2] dataset. This dataset currently holds 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).

We use two different timesteps in our experiments: 40ms (the resolution of [13]); and 16th-note, in which the input is divided into 16th-note frames based on the metrical grid. For the 16th-note timesteps, we use the metrical annotations from the A-MAPS dataset [26]. To downsample the acoustic prior for the 16th-note timesteps, we take the average of its original 40ms frames for the duration of each new frame. Before evaluation, we upsample our outputs back to 40ms timesteps, assigning each resulting frame the value of the corresponding output frame.

4.2 Metrics

We report both framewise and two versions of notewise precision (P), recall (R), and F-measure (F_1), each of which is averaged across all recordings in the test set. The frame-based metrics are standard as used in the MIREX Multiple-F0 Estimation task [1], comparing the output piano roll to the ground truth piano roll, using 40ms frames (after upsampling when using 16th-note timesteps). 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. We leave an analysis using proper notes for future work.

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. Additionally, we report both onset-only (On) and onset-offset (OnOff) notewise results. For On, a note is considered correct if its pitch is correct and its onset time is within 50ms of the ground truth, and for OnOff, we add the constraint that the offset is such that the note duration is within 20% of ground truth (or 50ms, whichever is biggest). Both are as described for note-tracking in [1], and we use `mir_eval` [18] to perform all calculations.

As argued in [12], the most relevant metrics are the notewise metrics. Indeed, a poor transcription system could still score high in terms of framewise F_1 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 [4, 8]. Thus, our main evaluation concentrates on onset metrics, and we discuss the OnOff metrics only in Section 5.4.

4.3 Configurations

Besides the two versions of our blending model described in Section 3.2 (WM and PM), we use a baseline blending model: a constant weight (CW) model, which always calculates $P_{p,t}$ 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 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+S, 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 5.2.

4.4 Baselines

We compare our models against that of [13], retrained with our training and validation sets. We threshold its output at a 0.5, setting all values ≥ 0.5 to 1 and all others to 0.

We also compare our model against a common HMM baseline [17] where each pitch is represented by a simple 2-state (on/off) HMM, run independently. Typically, the

[2] `http://piano-midi.de/`

Method	40ms timesteps						16th note timesteps					
	Framewise			On-Notewise			Framewise			On-Notewise		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
[13]	73.0	**65.5**	**68.3**	54.2	65.7	58.1	75.2	65.5	69.2	70.9	63.6	65.9
HMM	74.6	63.9	68.0	64.0	62.2	61.9	73.7	**69.4**	**70.7**	74.4	66.4	69.1
CW	73.7	64.9	68.2	61.2	63.3	61.1	76.6	61.6	67.3	76.1	55.7	63.0
CW+S	73.9	64.8	68.2	61.3	63.3	61.3	76.6	61.5	67.3	74.8	55.9	62.6
WM	73.0	64.8	67.9	63.0	60.8	61.0	77.2	62.6	68.0	75.6	61.4	66.7
WM+S	75.1	62.3	67.3	**67.6**	60.2	**62.8**	**77.8**	61.2	67.5	**79.4**	58.5	66.0
PM	**81.8**	50.8	60.8	57.0	**66.5**	59.9	77.2	63.9	69.1	72.4	66.4	68.3
PM+S	79.7	57.4	65.6	61.4	65.6	62.6	77.6	64.2	69.3	76.8	**68.7**	**71.7**

Table 2. 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.

HMM raises precision and lowers recall, removing short spurious notes from the output. In [17], 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 learn the transition probabilities with Bayesian Optimization (BO) [16] to maximize the notewise F_1 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. 16th-note 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 framewise and notewise F_1 for both timesteps compared to the MLE-trained HMM (MLE results omitted).

4.5 Training

The MLM is trained using the Adam optimizer [14] 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 timesteps. 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_{GT} linearly from 1 to 0.7 over 500 epochs. Validation is done using a fixed value of $p_{GT} = 0.7$, and we use early stopping once the schedule is finished (after 500 epochs).

The blending model is trained on the validation set to maximize On-notewise F_1. 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 Δ_{min}. Bayesian Optimization for 200 iterations is used to search for the values of Δ_{min} and $hist$ (up to 10 for 16th-note timesteps and up to 50 for 40ms timesteps), and how many hidden layers l to use (1–4 of size 5).

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. RESULTS

Full framewise and On-notewise results are in Table 2. Overall, we can see that for 40ms timesteps, PM+S and WM+S outperform all other models in the On-notewise F_1 ($p < 10^{-3}$ with a paired t-test, which we use for all significance tests), but WM+S does not significantly outperform PM+S. For 16th note timesteps, PM+S is significantly better than all other models for On-notewise F_1 ($p < 10^{-6}$). The baselines achieve the best framewise results—HMM for 16th-note ($p = 0.021$ over PM+S) and [13] for 40ms (not significant). This makes sense, as our blending models are optimised for On-notewise F_1. Moreover, our MLM is designed to take advantage of knowledge of musical structure, which is more clear at the note level.

In the following sections, we investigate the impact of each component of our system: the timestep (5.1), scheduled sampling (5.3), and the blending model (5.2). The OnOff-notewise metrics are presented in Section 5.4.

5.1 Timestep

It is clear that using 16th-note timesteps improves the results (framewise for the baselines and On-notewise for all methods). Similar results were seen in [27], which showed that the improvement was mainly due to quantisation of the output. Here, the increase in performance of [13] is due entirely to this quantisation. However, when performing the same quantisation procedure to the output of the 40ms-timestep PM+S, we see framewise and On-notewise F_1 of only 63.6 and 62.9 respectively, significantly worse than PM+S with a 16th-note timestep ($p < 10^{-6}$ for both). PM also sees a significant improvement when using a 16th-note timestep directly ($p < 10^{-8}$ for both), as well as WM+S, but for On-notewise F_1 only ($p = 0.01$). Weaker or insignificant effects are seen for our other models, both framewise and On-notewise. This shows that for our MLM, much of its improvement with 16th-note 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. In the following sections, therefore, we concentrate on the results with the 16th-note timestep.

5.2 Blending model

The adaptability of WM and PM clearly allow them to outperform CW, and the wider output range of PM leads to improvement over WM ($p = 0.018$ for WM over CW without scheduled sampling, $p < 10^{-6}$ for all other pairs), although WM's precision is greater. For the framewise metrics, a similar pattern is seen, though the effect is weaker ($p < 0.001$ for PM over WM, $p = 0.048$ from WM over CW, and not significant for WM+S over CW+S).

To see whether our improvement comes from the MLM or simply from the use of the blending model, we evaluate the PM-A configuration. This model achieves an On-notewise F_1 of 65.1 with a 16th-note timestep, significantly worse than PM+S ($p < 10^{-9}$), which 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.

5.3 Scheduled sampling

We can see that PM+S performs significantly better overall than PM for On-notewise ($p < 10^{-7}$), but not framewise F_1 ($p = 0.50$). Looking at the results in a piecewise fashion leads to an interesting conclusion about where that improvement comes from. In Figure 2, we plot the notewise F_1 of [13] (x-axis, a proxy for the noisiness of the input) against the increase in On-notewise F_1 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 ($p = 0.02$), but with high variance ($R^2 = 0.09$). Overall, we can conclude that scheduled sampling does indeed lead to improved performance with noisy inputs.

5.4 Onset-offset Evaluation

Table 3 presents the OnOff-notewise results for the two baselines and our overall best performing model (PM+S), where it can be seen that our model significantly outperforms the other systems ($p < 10^{-5}$ for both timesteps).

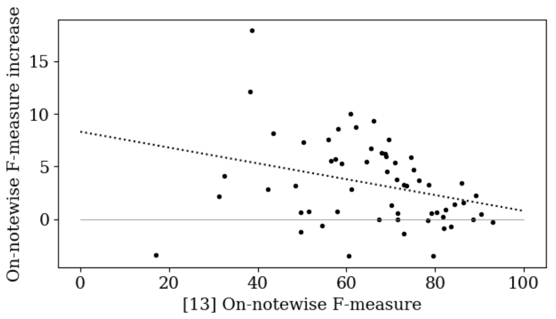

Figure 2. Increase of On-notewise F_1 of PM+S over PM plotted against On-notewise F_1 of [13] for each piece. Dotted line shows linear correlation ($p = 0.02$, $R^2 = 0.09$).

Method	40ms timestep			16th-note timestep		
	P	R	F_1	P	R	F_1
[13]	31.8	37.7	33.8	41.0	37.3	38.5
HMM	37.8	36.5	36.5	41.3	37.5	38.8
PM+S	**41.5**	**43.2**	**41.8**	**47.7**	**43.3**	**45.0**

Table 3. Results using the OnOff-notewise metrics. PM+S uses our prior model with scheduled sampling.

This is promising, and it seems that the MLM might have learned some rhythmic components of musical structure.

6. CONCLUSION

In this paper, we have presented a system for converting a posteriogram 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 than thresholding the posteriogram, as well as post-processing it with a new strong baseline HMM. We have further shown that (1) scheduled sampling helps the MLM perform better in the case of noisy inputs, and (2) the use of a 16th-note timestep allows the MLM to learn musical structures better than with a 40ms timestep.

To that end, in future work, we intend to investigate the use of noisy 16th-note labels from a beat-tracking system, rather than the ground truth labels that we have used here. Furthermore, we will perform a systematic ablation study for the input features of the blending model. We also intend to analyze our results in a more qualitative fashion in future work with listening tests. Our own subjective conclusions are that our system does often produce more musical results than the baselines, but a proper listening test would show more objectively whether that is the case throughout the test set, and what aspects of the resulting piano rolls become more musical in which cases. In the meantime, we provide some examples of our model's performance as supplementary material [3], along with the code to reproduce our experiments [4].

[3] c4dm.eecs.qmul.ac.uk/ycart/ismir19.html
[4] github.com/adrienycart/MLM_decoding

7. ACKNOWLEDGEMENTS

Thanks to Dr. Eita Nakamura for his help on this project. A. Ycart is supported by a QMUL EECS Research Studentship. A. McLeod and K. Yoshii are supported in part by JST ACCEL No. JPMJAC1602, JSPS KAKENHI No. 16H01744 and No. 19H04137, and the Kyoto University Foundation. E. Benetos is supported by UK RAEng Research Fellowship RF/128.

8. REFERENCES

[1] 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)*, pages 315–320, 2009.

[2] 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.

[3] 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*, pages 1171–1179, 2015.

[4] 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)*, pages 255–261, 2016.

[5] 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.

[6] F.J. Cañadas Quesada, N. Ruiz Reyes, P. Vera Candeas, J.J. Carabias, and S. 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.

[7] Dorian Cazau, Yuancheng Wang, Olivier Adam, Qiao Wang, and Grégory 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)*, pages 523–530, 2017.

[8] 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.

[9] Arnaud Dessein, 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)*, pages 489–494, 2010.

[10] 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.

[11] Graham Grindlay and Daniel P. W. Ellis. Multi-voice polyphonic music transcription using eigeninstruments. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 53–56, 2009.

[12] 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.

[13] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. *17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 475–481, 2016.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[15] 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.

[16] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

[17] 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.

[18] 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.

[19] Matti P. Ryynanen and Anssi Klapuri. Polyphonic music transcription using note event modeling. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 319–322. IEEE, 2005.

[20] Rodrigo Schramm, Andrew McLeod, Mark Steedman, and Emmanouil Benetos. Multi-pitch detection and voice assignment for a cappella recordings of multiple singers. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 552–559, 2017.

[21] 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*, 24(5):927–939, 2016.

[22] David Temperley. A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research*, 38(1):3–18, March 2009.

[23] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[24] Qi Wang, Ruohua Zhou, and Yonghong Yan. Polyphonic piano transcription with a note-based music language model. *Applied Sciences*, 8(3), 2018.

[25] Adrien Ycart and Emmanouil Benetos. A study on LSTM networks for polyphonic music sequence modelling. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 421–427, 2017.

[26] Adrien Ycart and Emmanouil Benetos. A-MAPS: Augmented MAPS dataset with rhythm and key annotations. In *ISMIR Late Breaking and Demos Papers*, 2018.

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