POLYPHONIC MUSIC SEQUENCE CLASSIFICATION WITH LSTM NETWORKS

Adrien Ycart **Emmanouil Benetos**
Centre for Digital Music, Queen Mary University of London
`{a.ycart/emmanouil.benetos}@qmul.ac.uk`

ABSTRACT

We describe our submission for the symbolic polyphonic Patterns for Prediction MIREX task (subtask 2). Our system is a simple LSTM model trained for prediction. We investigate more specifically the use of various newly proposed metrics as training losses and classification criteria. We also propose to train our model in a multi-task fashion for both prediction and classification.

1. INTRODUCTION

We aim to evaluate the ability of a simple music language model (MLM), similar to the one presented in [3], to discriminate between real and fake polyphonic music sequences. The general idea is that our model should be better at modelling the data it was trained on, in our case, real music sequences, than the fake continuations. The model should thus yield better results, according to some metric, when evaluated on the real data than on the fake. We focus our proposal on the comparison of various metrics as training losses and classification criteria. The metrics are defined in Section 3. A summary of the submitted combinations is given in Table 1.

2. SYSTEM OVERVIEW

2.1 Input format

Out model takes as input a piano-roll matrix, in other words, a binary matrix M of shape $N \times T$, such that $M_{p,t} = 1$ if and only if pitch p is active at timestep t. In particular, we make no distinction between held and repeated notes. Here, we choose $N = 128$, to be able to represent every MIDI pitch. Each piano roll is the concatenation of a priming piano roll and a continuation piano roll (either real or fake).

2.2 Model Architecture

We use a simple configuration of the most widely-used RNN unit: the Long Short-Term Memory (LSTM) [1]. We use a single layer with 128 inputs and 256 hidden units, followed by a dense layer with 128 outputs. We use sigmoid activations for the dense layer. We call $\hat{M}$ the non-binary matrix corresponding to the outputs of the network.

2.3 Training setups

2.3.1 *Prediction only*

In this setup, we train our model on a simple prediction task, as commonly done in natural language processing. Let M_t be the 128-vector corresponding to time frame t of M, given $(M_t)_{t\in[\![0,n-1]\!]}$, the model tries to predict M_n. Given a metric $\mathcal{M}_{pred}$, the network minimises the loss $\mathcal{L}_{pred} = \mathcal{M}_{pred}(M_{real}, \hat{M}_{real})$ where M_{real} is a real piano roll. Here, the network simply tries to model the real data, and doesn't use the fake data.

2.3.2 *Prediction + Classification*

In this setup, we train our model in a multi-task fashion: the model has to predict the next frame of the real data, and also to discriminate correctly between real and fake data. Let $\mathcal{M}_{class}$ be a metric, the classification output C is defined as a vector of size 2 such that : $C_0 = \mathcal{M}_{class}(M_{real}, \hat{M}_{real})$ and $C_1 = \mathcal{M}_{class}(M_{fake}, \hat{M}_{fake})$, where M_{real} and M_{fake} are a real and a fake piano roll respectively. Let $\mathbb{S}$ be the softmax function, we then define $\mathcal{L}_{class}$ as the cross-entropy between $\mathbb{S}(C)$ and the labels $[1, 0]$. We then use $\mathcal{L}_{pred} + \mathcal{L}_{class}$ as training loss.

3. METRICS

We define a set of metrics that can be computed on each piano roll. These can then be used either as training objective or as classification criterion. All the newly proposed metrics (Sections 3.2 to 3.7) will be defined and investigated in [4].

3.1 Cross-entropy

We use the usual definition of cross entropy. Let $\hat{M}_t = \left(\hat{M}_{t,p}\right)_{p\in[\![0,87]\!]}$ be the sigmoid output of our network at timestep t, and $M_t = \left(M_{t,p}\right)_{p\in[\![0,87]\!]}$ the binary targets at the same timestep. The cross entropy $\mathcal{H}(M_t, \hat{M}_t)$ between the vectors M_t and $\hat{M}_t$ is defined as:

$$\mathcal{H}(M_t, \hat{M}_t) = -\sum_{p\in[\![0,127]\!]} M_{t,p} \log(\hat{M}_{t,p}) + (1 - M_{t,p}) \log(1 - \hat{M}_{t,p}) \quad (1)$$

Submission ID	Training setup	Prediction loss	Classification loss	Classification criterion
YB1	Prediction only	$\mathcal{H}$	-	$\mathcal{H}$
YB2	Prediction only	$\mathcal{H}$	-	$\mathcal{H}_{tr}$
YB3	Prediction only	$\mathcal{H}$	-	$\mathcal{S}$
YB4	Prediction only	$\mathcal{S}$	-	$\mathcal{S}$
YB5	Prediction + Classification	$\mathcal{H}$	$\mathcal{H}_{tr}$	$\mathcal{H}_{tr}$

Table 1. Summary of the submitted systems. The definition of the losses and criteria is given in Section 3.

The cross entropy $\mathcal{H}(M, \hat{M})$ is defined as the average of the cross entropy across timesteps.

3.2 Transition Cross entropy

A good model must be able to successfully predict transitions, not only continuations of notes.. In order to evaluate this ability separately, we compute the cross entropy only on frames where there is a transition.

For a given M, let Tr be the subset of $[\![1, T-1]\!]$ such that:

$$t \in Tr \Leftrightarrow M_t \neq M_{t-1} \tag{2}$$

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

$$\mathcal{H}_{tr} = \frac{1}{|Tr|} \sum_{t \in Tr} \frac{\mathcal{H}(M_t, \hat{M}_t)}{d(t)} \tag{3}$$

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.

3.3 Steady-state Cross entropy

Transition cross entropy evaluates the ability of an MLM in the 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 :

$$\mathcal{H}_{ss} = \frac{1}{T - 1 - |Tr|} \sum_{t \in [\![1, T-1]\!] \setminus Tr} \mathcal{H}(M_t, \hat{M}_t) \tag{4}$$

Here, we do not normalise by the number of active note 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.

3.4 Pitch-profile Cross entropy

We introduce the pitch-profile cross entropy. It basically assesses how relevant to the piece the erroneous outputs of the system are.

Usually, a scale is defined as a subset of the 12 notes of the chromatic scale. In particular, the scale of a piece is invariant across octaves. 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 pitch-profile of a piece $\mathrm{P}(t)$ is then defined as:

$$p \in \mathrm{P}(t) \iff \frac{1}{T} \sum_{t<T} M(t,p) > 0.05 \tag{5}$$

We subsequently define a scale vector $P(t)$ such that $P(t)_p = 1 \iff p \in \mathrm{P}(t)$. It has to be noted that we use the same $P(t)$ when evaluating the real and fake piano rolls, computed from the real one only.

We only want to evaluate how close to the scale the erroneous predictions are. Indeed, our focus here is to measure to what extent the false positives, despite being mistakes, make sense from a musical point of view. In order to get rid of the influence of the correct notes, we do not consider in the computation of the pitch-profile cross entropy the bins where the target is equal to 1. We call the ensemble of such bins B:

$$B = \{(p,t) \in [\![0, 127]\!] \times [\![1, T-1]\!] \mid M_{t,p} = 0\} \tag{6}$$

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 (p, t), the pitch-profile cross entropy is given as:

$$\mathcal{H}_{pp}(p,t) = -P(t)_p \log(\hat{M}_{t,p}) - (1 - P(t)_p) \log(1 - \hat{M}_{t,p}) \tag{7}$$

We then define the pitch-profile cross entropy for a piano roll as follows:

$$\mathcal{H}_{pp} = \frac{1}{|B|} \sum_{(p,t) \in B} \mathcal{H}_{pp}(p,t) \tag{8}$$

3.5 Transition-Pitch-profile Cross entropy

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 $B_t = \{p \in [\![0, 127]\!] \mid (t,p) \in B\}$. We then have:

$$\mathcal{H}_{pp,tr} = \frac{1}{\sum_{t \in Tr} |B_t|} \sum_{t \in Tr, p \in B_t} \mathcal{H}_{pp}(t,p) \tag{9}$$

3.6 Steady-State-Pitch-profile Cross entropy

By analogy with $\mathcal{H}_{ss}$(see Section 3.3) , we also define the steady-state-pitch-profile cross entropy:

$$\mathcal{H}_{pp,ss} = \frac{1}{\sum_{t \notin Tr} |B_t|} \sum_{t \notin Tr, p \in B_t} \mathcal{H}_{pp}(t,p) \qquad (10)$$

3.7 Proposed unified metric

All the proposed metrics capture different aspects of the performance of an MLM, similarly to precision and recall for classification: $\mathcal{H}_{tr}$ focuses on transitions, $\mathcal{H}_{ss}$ on continuations, while $\mathcal{H}_{pp}$, $\mathcal{H}_{pp,tr}$ and $\mathcal{H}_{pp,ss}$ focus on tonality. Optimising a model for only one of them can lead to undesired behaviour, but they all capture some good aspects of a model. In order to have one single measure to indicate the fitness of a model, similarly to F-measure, we propose a unified metric that combines the above proposed metrics.

The metric we propose, for which lower is better, is defined as:

$$\mathcal{S} = \sqrt{(\mathcal{H}_{tr} + \mathcal{H}_{ss})(\mathcal{H}_{pp,tr} + \mathcal{H}_{pp,ss})} \qquad (11)$$

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 weighting. Here, 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 (as is the case here), they have a much higher weight with this metric than with $\mathcal{H}$, which helps put the emphasis on the difficult parts. Similarly, summing $\mathcal{H}_{pp,tr}$ and $\mathcal{H}_{pp,ss}$ allows to evaluate all bins with respect to the pitch-profile, but gives more weight to the rarer type of frames.

4. EXPERIMENTS

4.1 Dataset

To train our network, we use the provided dataset: PPDD-Sep2018_sym_poly_medium. This dataset contains 1000 triplets (priming sequence, real continuation, fake continuation). We split it into training and validation subsets(80% and 20% of examples respectively). We test our systems on 1000 randomly chosen examples from the dataset PPDD-Sep2018_sym_poly_large (we use the same test set to compare all configurations). Although there is no guarantee that there is no overlap with the training set, we assume that this is unlikely given the way the datasets were built. We give more importance to having enough test samples to avoid sample bias.

The data is converted to piano-roll format, using a timestep of a 48th note (a 12th of a quarter note), which is the smallest common multiple of all the durations present in the dataset. All MIDI channels are collapsed into one single matrix. For each example, we have one real piano roll and one fake piano roll.

To improve transposition invariance, we also augment our dataset by transposing every training example up to 2 semitones up or down. For each example, the real and fake continuations are transposed identically.

Submission ID	Accuracy (%)
YB1	42.6
YB2	66.0
YB3	64.0
YB4	61.8
YB5	**69.6**

Table 2. Preliminary results, with best values in bold.

4.2 Training setup

We use the Adam optimiser [2] to train our model, with a learning rate of 0.01. We also use early stopping, such that if the loss evaluated on the validation set hasn't decreased for 15 epochs, we stop training and keep the best model so far.

4.3 Preliminary results

We run a range of experiments with the various configurations presented in Table 1. YB1 is the default configuration. With YB2 and YB3, we investigate other metrics as classification criterion, using the same trained model. With YB4, we investigate the effect of training our model with $\mathcal{S}$ instead of the usual $\mathcal{H}$. Preliminary experiments showed that the best results were obtained with YB2, so we try to push performance further by adding a classification training objective, using $\mathcal{H}_{tr}$ as classification function.

In each configuration, for each example, we compute $v_{real} = \mathcal{M}(M_{real}, \hat{M}_{real})$ and $v_{fake} = \mathcal{M}(M_{fake}, \hat{M}_{fake})$, where $\mathcal{M}$ is the classification criterion. We count a correct classification when $v_{real} < v_{fake}$. The reported accuracy is the percentage of correct classifications. The results are summarised in Table 2.

4.4 Output normalisation

For the purpose of this submission, we need to have v_{real} and v_{fake} between 0 and 1. We thus divide these two values by their sum $v_{real} + v_{fake}$. As M_{real} and M_{fake} only differ by the continuation sections, they tend to yield very similar values. To spread them out, we normalise them so the biggest distance between v_{real} and v_{fake} in the dataset is 1. Eventually, we have (with $v \in \{v_{real}, v_{fake}\}$):

$$output = \frac{\frac{v}{v_{real}+v_{fake}} - 0.5}{max(\frac{|v_{real}-v_{fake}|}{v_{real}+v_{fake}})} + 0.5$$

It can happen that $\mathcal{S}(M, \hat{M})$ is ill-defined, for instance when there are not steady-state frames. In this case, we count it as an incorrect classification, and we output nominal values $v_{real} = v_{fake} = 0.5$

5. REFERENCES

[1] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

[3] 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)*, pages 421–427, 2017.

[4] Adrien Ycart and Emmanouil Benetos. Defining New Metrics for Music Language Modelling. *In Preparation*, 2019.