Technical Report - Musical Features for Automatic Music Transcription Evaluation

Adrien Ycart, Lele Liu, Emmanouil Benetos, Marcus T. Pearce

1 Introduction

Automatic Music Transcription (AMT) is most often evaluated with metrics that penalise all mistakes equally (see [1] for a description of common AMT metrics). However, all mistakes are not equally salient to human listeners: for instance, out-of-key false positives tend to be very noticeable. In order to get more insights into the types of mistakes made by a system, we define new, lower-level metrics that can be computed on pairs (target, AMT output). We aim to define these metrics so that they capture musical aspects that are reported as important by human raters, and mistakes commonly made by AMT systems.

These metrics are defined with polyphonic piano music transcription as the main application. However, most of them are more general and can be applied in other contexts with minor modifications.

In what follows, we describe the notations we use throughout this document in Section 2.

2 Data format

The AMT output and target are usually represented either as a piano roll or a list of notes.

A piano roll is a $N_p \times T$ binary matrix, where T corresponds to the number of timesteps, and N_p to the number of considered pitches (88 in the case of piano). We notate the estimated and ground-truth piano rolls as $\hat{M}$ and M respectively. Roughly, $M[p,t] = 1$ if and only if pitch p is active at timestep t. We notate M_t the binary N_p-vector, which corresponds to the t-th frame of M.

The AMT output and target can also be represented as a list of notes. We notate them $\hat{N}$ and N respectively. They are lists of potentially different lengths $\hat{n}$ and n respectively. Each element of N is a tuple (s, e, p, v) where s and e are the start and end times, p is the MIDI pitch, and v is the original velocity of the note. Each element of $\hat{N}$ is a tuple (s, e, p) (same, without velocity, as it is often not estimated by AMT systems).

3 Features

3.1 Benchmark Framewise Precision, Recall, F-measure

These framewise metrics are computed on piano-roll outputs, with a frame duration of 10ms. We notate for each frame t the number of true positives, false positives and false negatives as respectively, $TP(t)$, $FP(t)$ and $FN(t)$. A true positive is counted when $M[p,t] = 1$ and $\hat{M}[p,t] = 1$. We define the framewise Precision, Recall and F-measure as:

$$\mathcal{P} = \frac{\sum_{t=0}^{T} TP(t)}{\sum_{t=0}^{T} TP(t) + FP(t)} \tag{1}$$

$$\mathcal{R} = \frac{\sum_{t=0}^{T} TP(t)}{\sum_{t=0}^{T} TP(t) + FN(t)} \tag{2}$$

$$\mathcal{F} = \frac{2 \cdot P \cdot R}{P + R} \tag{3}$$

3.2 Benchmark Onset-only Notewise Precision, Recall, F-measure

These notewise metrics are computed on lists of note outputs. We count a true positive when a note $(\hat{s}, \hat{e}, \hat{p})$ is detected and there is a reference note (s, n, p) such that:

$$|\hat{s} - s| < 50ms \wedge \hat{p} = p \tag{4}$$

Each reference note must be matched to at most one estimated note.

We then compute the Precision, Recall and F-Measure as above. The maximum matching between target and output is computed using `mir_eval`[1].

In what follows, we use this definition to determine whether a detected note is a true positive, unless stated otherwise, as it was shown in [7] that onset-only F-measure is the benchmark evaluation metric that correlates best with human perception (in the case of piano). When used for other instruments, the onset-offset definition might be preferred.

3.3 Benchmark Onset-offset Notewise Precision, Recall, F-measure

These metrics are the same as above, with the difference that a true positive is counted when:

$$|\hat{s} - s| < 50ms \wedge \hat{p} = p \wedge |\hat{e} - e| < max(50ms, 0.2 * (e - s)) \tag{5}$$

[1] `https://github.com/craffel/mir_eval`

We then compute Precision, Recall and F-Measure as above.

3.4 Number of mistakes in highest and lowest voice

The general idea is that very often, mistakes in the melody or the bassline are more salient than mistakes in middle voices. We use the highest and lowest voice as a proxy for the melody and the bassline, respectively. The highest and lowest voices can be defined both framewise or notewise.

Usually, the ground truth is defined so that it corresponds as much as possible to what is contained in the audio signal. In particular, when the sustain pedal is used, the offsets of notes are usually extended to correspond to the actual duration for which they sound. When defining the highest and lowest voice, we choose to not take the sustain pedal into account, in order to stick as much as possible to the original partition, and avoid excessive overlapping of notes. We assume that this is accessible, as most of the time, the sustain pedal is given as an external MIDI control-change parameter, that can either be taken into account or not.

3.4.1 Framewise

The highest voice H_M is a time series such that:

$$H_M(t) = \max_{p \in [0, N_p[} (p \mid M[p,t] = 1) \tag{6}$$

If M_t is all zeros, we define by default $H_M(t) = -1$.

- A true positive is counted for each (p,t) such that $p = H_M(t)$ and $H_M(t) \neq -1$ and $\hat{M}[p,t] = 1$.
- A false negative is counted for each (p,t) such that $p = H_M(t)$ and $H_M(t) \neq -1$ and $\hat{M}[p,t] = 0$.
- A false positive is counted for each (p,t) such that $\hat{M}[p,t] = 1$ and $p > H_M(t)$. We count all false positives above the highest pitch in the target.

With these definitions, we can compute Precision, Recall, and F-measure as before.

The lowest voice can be defined similarly.

3.4.2 Notewise

The highest voice H_N is a list of notes such that for all (s,e,p) in H_N:

$$\exists t \in [s,e] \mid \forall (s',e',p') \in N, t \notin [s',e'] \vee p' < p \tag{7}$$

In other words, it is the set of notes that are the highest sounding note at some point in time. In order to account for cases when, for instance, a chord is slightly arpeggiated and a middle note is played before the highest note,

we include a minimum duration d_H for which the note has to be the highest sounding one:

$$\exists t_1, t_2 \in [s, e] \mid t_2 - t_1 > d_H \wedge \forall (s', e', p') \in N, [t_1, t_2] \cap [s', e'] = \emptyset \vee p' < p \quad (8)$$

- A true positive is counted for each note that is a true positive and that is matched to a note in H_N.
- A false negative is counted for each note in H_N that is left unmatched.
- A false positive is counted for each note (s, e, p) in $\hat{N}$ that is left unmatched and such that

  $$\exists t \in]s, e[, \forall (s', e', p') \in N, t \notin]s', e'[\vee p' < p$$

  or with a threshold:

  $$\exists t_1, t_2 \in [s, e] \mid t_2 - t_1 > d_H \wedge \forall (s', e', p') \in N, [t_1, t_2] \cap [s', e'] = \emptyset \vee p' < p$$

  We count all false positives above the highest pitch in the target.

We set $d_H = 0.5$, this value is set heuristically, in accordance with the usual threshold for benchmark notewise metrics.

With these definitions, we can compute Precision, Recall, and F-measure as before.

The lowest voice can be defined similarly.

3.5 Loudness of false negatives

The idea is that a missed note will be less noticed if it was played very softly in the input than if it was very loud originally. It will be even less noticed if there are some louder notes played at the same time.

Similar metrics could be defined with false positives, but since most current AMT systems do not estimate note velocities, we do not take them into account.

We define two such metrics: the normalised false negative loudness, and the false negative loudness ratio.

3.5.1 Normalised false negative loudness

For each false negative, we normalise its loudness by the average loudness in a 2-seconds window centered on the onset of the false negative. In other words, for a given false negative (s, e, p, v) in N define V as:

$$V = \{(s', e', p', v') \mid |s - s'| < d_L\} \quad (9)$$

where d_L is set as 1 second. we then compute the normalised loudness as:

$$\text{NormLoud} = \frac{v \times |V|}{\sum_{(s', e', p', v') \in V} v'} \quad (10)$$

where $|V|$ denotes the cardinal of V. We then compute the average normalised loudness for all false positives. It has to be noted that V is never empty; it always contains at least (s, e, p, v). When (s, e, p, v) is the only note in V, the ratio is equal to 1.

3.5.2 False negative loudness ratio

We also compute the ratio between the missed note velocity and the loudest note sounding in the ground truth at the time of the attack of the missed note. This ratio is necessarily smaller than 1. In particular, it is equal to 1 when the missed note is the only one at that time. We could then average this ratio for all the false negatives.

To do so, we assume an exponential decay of the amplitude of notes, to reflect cases where a loud note is held while other notes are played. We use an exponential decay rate, applied to the velocities directly. We stop the decay after 1 second, to avoid notes fading out completely. Given $a(p)$ the decay rate of a note given its pitch, we define the time varying velocity $v(t)$ of a note (s,e,p,v) as:

$$v(t) = \begin{cases} ve^{-a(p)(t-s)} & \text{if } t \in [s, s+1] \\ ve^{-a(p)} & \text{if } t \in [s+1, e] \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

The decay rate is obtained from the values presented in Figure 1, taken from [2]. For each pitch, we average the decay rates across velocities (this has little influence for lower pitches). Then, we do a linear regression on the averaged decay rates, in order to have smoother, less piano-dependent decay rates estimates. Using that, we can then compute the velocity of notes through time. The formula we obtain for $a(p)$ is:

$$a(p) = 0.050532 + 0.021292 * p \tag{12}$$

The loudness ratio of a false negative (s, e, p, v) is then defined as:

$$\text{LoudRatio} = \frac{v}{\max\limits_{(s',e',p',v') \in N, t \in [s-d_R, s+d_R]} v'(t)} \tag{13}$$

where d_R is a parameter that we set to 50ms.

3.6 Out-of-key false positives

The idea is that out-of-key inserted notes are particularly salient. However, we do not want to rely on key annotations to evaluate that. Instead, we rely on a pitch profile that is automatically computed from the target. We propose two versions of this metric, one based on a binary pitch profile, and one on a non-binary pitch profile.

The following definitions assume that the tonality is constant throughout the considered music excerpt. Behaviour is undefined if that is not the case.

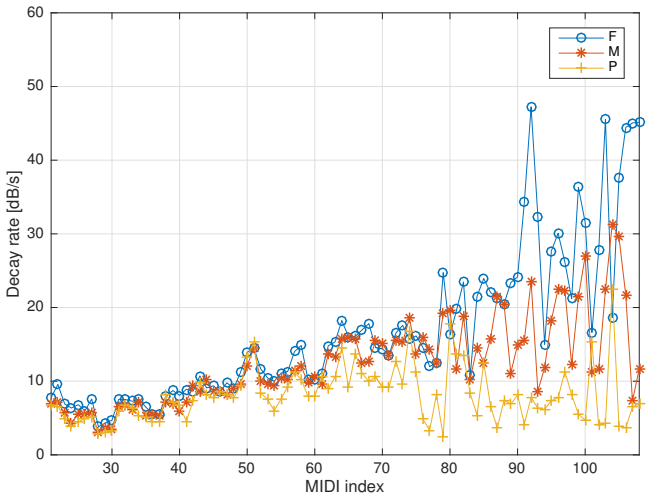

Figure 1: Decay rates per note, for various velocities (taken from [2]).

3.6.1 Binary pitch profile

We define the binary pitch profile as the set of pitch classes that are active more than 10% of the time in the target. This threshold is set heuristically. Generally-speaking, the higher threshold, the more notes will be considered as out-of-key. It remains to be seen to what extent varying this threshold influences the metrics.

More precisely, we define an active-pitch-class time series $A_p(t)$ such that:

$$A_p(t) = 1 \iff \exists (s, e, q) \in N \mid q \equiv p \pmod{12} \wedge t \in [s, e] \tag{14}$$

$A_p(t)$ is defined using the target without pedal, similarly to Section 3.4.

We write P_b the set of in-key pitches such that:

$$p \in P_b \iff p \equiv q \pmod{12} \wedge \frac{1}{T} \sum_{t \in [0,T[} A_q(t) > 0.1 \tag{15}$$

We write $FP_{\hat{N}}$ the total number of false positive notes in $\hat{N}$, $FP_{P_b,\hat{N}}$ the number of out-of-key false positives (i.e. the false positives such that $p \notin P_b$), and $|\hat{N}|$ the total number of notes in $\hat{N}$. We then define two ratios:

$$\mathcal{O}_{b,t} = \frac{FP_{P_b,\hat{N}}}{|\hat{N}|} \text{ and } \mathcal{O}_{b,p} = \frac{FP_{P_b,\hat{N}}}{FP_{\hat{N}}} \tag{16}$$

$\mathcal{O}_{b,t}$ is the proportion of out-of-key mistakes among all detected notes (and is thus correlated with notewise recall), while $\mathcal{O}_{b,p}$ is the proportion of out-of-key mistakes among all false positives.

3.6.2 Non-Binary pitch profile

For each pitch class q, we define its fitness to the tonality $F(q)$ as the proportion of the time this pitch class is active:

$$F(q) = \frac{1}{T} \sum_{t \in [0,T[} A_q(t) \tag{17}$$

Then, we define the key-disagreement of a false positive (s, e, p) as: $1 - F(q)$ for q such that $q \equiv p \pmod{12}$

We then compute 2 values: $\mathcal{O}_{b,t}$ the average key-disagreement of false positives divided by the average key-disagreement of all detected notes, and $\mathcal{O}_{b,p}$ the un-normalised average key-disagreement of false positives.

3.7 Specific Pitch Errors

A common type of AMT mistake is to have false positives in specific pitch intervals compared to ground-truth notes: semitone errors (neighbouring notes), octave errors (first partial), and 19 semitone errors (second partial). We define errors for specific pitches p_e with $p_e \in \{1, 12, 19\}$. We define these metrics both framewise and notewise.

3.7.1 Framewise

$\hat{M}[p, t]$ is a specific pitch error if and only if the following conditions are all true:

- $\hat{M}[p, t] = 1$
- $M[p, t] = 0$
- $M[p - p_e, t] = 1 \vee M[p + p_e, t] = 1$
- $\forall i \in [t - \delta, t[, M[p, i] = 0$

where δ is a parameter that we set heuristically to 50ms, in accordance with the threshold used for benchmark notewise metrics. The last condition is to ensure that erroneous continuations of correct notes are not penalised if there was also a target note p_e semitones apart right after.

For $p_e = 19$, we only consider ground-truth notes 19 semitones below, as second partial mistakes usually only happen above the ground truth notes. The third condition becomes simply: $M[p - p_e, t] = 1$

We then compute two different ratios: let N_s be the number of specific pitch errors, N_f the number of frames, and N_e the total number of false positives:

$$\mathcal{S}_{p_e,f,t} = \frac{N_s}{N_f} \text{ and } \mathcal{S}_{p_e,f,p} = \frac{N_s}{N_e} \tag{18}$$

The first is correlated to $\mathcal{P}$, while the second can be biased when $\mathcal{P}$ is very high (an output with only 1 error that happens to be a specific pitch error will have $\mathcal{S}_{f,p} = 1$)

3.7.2 Notewise

A note (s, e, p) is a specific pitch false positive if:

- (s, e, p) is a false positive
- $\exists (s', e', p') \in N \mid |p - p'| = p_e \wedge \frac{\min(e,e') - max(s,s')}{e-s} > 0.8$

The last condition boils down to saying that there is a ground truth note p_e semitones apart from (s, e, p) that overlaps with (s, e, p) for 80% of (s, e, p)'s duration. The value 80% was set heuristically, and echoes the benchmark onset-offset notewise metrics condition requiring that the offset of a note is within 20% of its duration.

For $p_e = 19$, we only consider ground truth notes 19 semitones below (s, e, p), for the same reason as above.

Similarly as framewise metrics, we compute 2 ratios: the proportion of specific pitch mistakes among all detected notes, and among false positives.

3.8 Repeated and merged notes

Another common type of mistake in AMT is to have repeated (i.e. fragmented) notes, or incorrectly merged notes.

3.8.1 Repeated notes

A note $(s, e, p) \in \hat{N}$ is counted as a repeated note when it fulfills the following conditions:

- It is an Onset-only false positive
- $\exists (s', e', p') \in N \mid p = p' \wedge \frac{\min(e,e') - max(s,s')}{e-s} > 0.8$
- $\exists (s'', e'', p'') \in \hat{N}$ such that:
  - $(s'', e'', p'') \neq (s, e, p)$
  - $p'' = p'$
  - $\frac{\min(e'',e') - max(s'',s')}{e''-s''} > 0.8$
  - $e'' < s$

Put more simply, a note is considered as a repeated note if it is a false positive, if it overlaps with a ground-truth note, and if there is another previous detected note that overlaps with the same ground-truth note.

We can then compute either the proportion of repeated notes among false positives, or among all notes, as with the previous metrics. Once again, for both of these metrics, the threshold 80% was set heuristically, and echoes the benchmark onset-offset notewise metrics condition requiring that the offset of a note is within 20% of its duration.

3.8.2 Merged notes

A note $(s, e, p) \in N$ is counted as a merged note when it fulfills the following conditions:

- It is an Onset-only false negative
- $\exists (s', e', p') \in \hat{N} \mid p = p' \wedge \frac{\min(e,e') - max(s,s')}{e-s} > 0.8$
- $\exists (s'', e'', p'') \in N$ such that:
  - $(s'', e'', p'') \neq (s, e, p)$
  - $p'' = p'$
  - $\frac{\min(e'',e') - max(s'',s')}{e''-s''} > 0.8$
  - $e'' < s$

A note is thus considered as a merged note if it is a false negative, if it overlaps with a detected note, and if there is another previous ground-truth note that overlaps with the same detected note.

We can then compute either the proportion of merged notes among false negatives, or among all notes, as with the previous metrics.

3.9 Rhythm features

3.9.1 Rhythm histogram spectral flatness

Rhythm is an important aspect of music. We thus define a metric to account for rhythmic imprecision as follows. We define O and $\hat{O}$ the (ordered) list of note onsets of N and $\hat{N}$ respectively, potentially with repetition. O is of same size as N.

Based on these, we compute inter-onset-intervals (IOI and $I\hat{O}I$) as the first derivative of O and $\hat{O}$ respectively. Let n be the number of notes:

$$\forall 0 \leq i < n-1, IOI(i) = O(i+1) - O(i) \tag{19}$$

We then compute a normalised histogram of the IOIs, with bins as follows: from 0 to 100ms, we use a bin size of 10ms, and from 100ms to 2s, we use a bin size of 100ms. Overall, we have 29 bins. We notate this list b, and the resulting histogram h_r. The spacing in bins is set heuristically; however, it could have a strong influence on the result. We leave it to future work to quantify that influence.

From this IOI, we compute its spectral flatness [4], which is defined as the ratio of the geometric mean of the histogram over its arithmetic mean: It is used usually on power spectra, and represents the peakiness of the spectrum. It is useful in our case, as quantised rhythms would give an IOI histogram with

only some non-zero bins, corresponding to specific note values, while rhythm imprecision would spread the values across several bins. We thus have:

$$S_f = \frac{1}{29} \sum_{0 \leq i < 29} \log(h_r[i]) - \log\left(\frac{1}{29} \sum_{0 \leq i < 29} h_r[i]\right) \tag{20}$$

In practice, we add $\epsilon = 10^-5$ to deal with 0 values in h_r.

We then compute the spectral flatness value for N and $\hat{N}$. We use as features the spectral flatness for $\hat{N}$ and the difference between the spectral flatness of $\hat{N}$ and N.

3.9.2 Rhythm dispersion

Another approach to attempt to characterise rhythmic deviations is to run K-means clustering [5] on the IOI set. Ideally, each cluster would correspond to one note value, with small variations due to tempo deviations and interpretation mostly. We can then assess the mean and standard deviation within each cluster.

Setting the number of clusters here is a tough problem. If we do not have enough clusters, one cluster might correspond to several note values, which would result in artificially high standard deviation. If we have too many clusters, we might end up with several clusters corresponding to the same note value, or in the extreme case, one cluster per note.

To determine the number of clusters and their initial centers, we compute a normalised IOI histogram on the target, similar to the previous, but with higher bin size: 20ms between 0 and 0.1s, and 200ms between 0.1 and 2s. We then choose as initial cluster centres all the peaks in the resulting h_r. Here again, the spacing in bins is set heuristically; however, it has to be noted that it might have a strong influence on the number of clusters.

We first run K-means clustering on the target IOI set. After convergence, we use the resulting cluster means as initial values to run K-means clustering on the estimated IOI set. We then compute the distance between cluster means for the estimated and target IOI sets, and the relative difference between the cluster standard deviations for the estimated and target IOI sets. We use as feature the mean, maximum and minimum across clusters, for both the centre drifts and standard deviation differences.

3.9.3 Validating Rhythm features

We have seen in the experiments presented in [7] that these rhythmic features have a high importance when modelling perceptual ratings of AMT quality. In order to validate that these metrics do capture rhythm deviations, we run some experiments.

We use as target the AMT outputs for all the stimuli in presented in [7]. We use as outputs various modified versions of these same MIDI files, by order of rhythm regularity (high to low):

Quant-constant: Quantised MIDI files with 16th note precision, using a constant tempo equal to the average tempo over the whole segment (we use the A-MAPS tempo and beat annotations described in [6]);

Quant: Quantised MIDI files with 16th note precision, using a time-varying tempo, with the ground-truth 16th note positions;

Noisy-100: Add uniform noise in [-100ms,100ms] to the onset times;

Noisy-300: Add uniform noise in [-300ms,300ms] to the onset times.

We report in Table 1 the mean and standard deviation (std) of the rhythm features in each condition. It appears that the features behave generally as expected. The mean spectral flatness is lowest for Quant-constant, and highest for the Noisy configurations. Besides, the spectral flatness difference is negative for the quantised versions, and positive for the noisy versions. For the rhythm dispersion values, we see a similar trend: for quantised versions, the average change in std is negative, while it is positive for noisy versions. Moreover, the greater the noise, the greater the average change in std.

However, it appears that increasing the noise level does not change the mean spectral flatness of the outputs, which is kind of surprising. This might be due to the bin size we used: since we use bins of size 100ms between 100ms and 2s, small differences in noise might be hard to catch. Another possibility is that since we use short examples, in a lot of cases, histogram bins contain one single value. Adding noise to that value will change the bin it is counted in, but will not change the overall spectral flatness. This might also explain why the dispersion average drift also increases with the noise level: although the noise is centred on zero, it is likely applied to many clusters with one single, or few values in it, so the drift does not always cancel out on average within a cluster.

It also appears that the dispersion feature values are very similar for both quantised versions. This might be due to the fact that we use short segments, and that tempo variations are quite small usually, so there is probably little difference between the two quantised conditions.

3.10 Consonance measures

We choose 3 different consonance measures: one based on periodicity/inharmonicity, one based on partials interference, and one based on culture (statistical frequency in a corpus). These are computed using Peter Harrison's implementation[2]. In particular, we use the following features:

- `hutch_78_roughness` for partials interference
- `har_18_harmonicity` for periodicity
- `har_19_corpus` for culture.

[2]https://github.com/pmcharrison/incon

Feature	Quant-constant		Quant		Noisy-100		Noisy-300	
	mean	std	mean	std	mean	std	mean	std
Spectral Flatness Output	*-9.842*	0.857	-9.572	0.909	**-6.695**	0.814	-6.751	0.772
Spectral Flatness Difference	*-1.850*	0.973	-1.580	0.898	**1.298**	1.217	1.241	1.380
Dispersion Avg. std Change	*-0.010*	0.019	*-0.010*	0.017	0.024	0.020	**0.080**	0.038
Dispersion Min. std Change	*-0.032*	0.036	-0.029	0.041	-0.001	0.050	**0.037**	0.074
Dispersion Max. std Change	0.012	0.031	*0.009*	0.026	0.049	0.027	**0.125**	0.048
Dispersion Avg. Drift	*0.025*	0.022	*0.025*	0.026	0.038	0.032	**0.129**	0.047
Dispersion Min. Drift	*0.008*	0.009	*0.008*	0.009	0.015	0.013	**0.057**	0.045
Dispersion Max. Drift	*0.046*	0.050	*0.046*	0.059	0.065	0.068	**0.209**	0.095

Table 1: Feature means and standard deviation (std) across all stimuli, with 4 levels of rhythmic precision. Highest mean values are in bold, lowest mean values are in italic.

These 3 consonance measures were shown to correlate best with perceptual ratings of consonance [3].

We then compute these consonance measures on the output and target piano rolls, using an *event* timestep: one timestep per new onset or offset. The above features are undefined for silence, we thus do not take them into account in the computations. We then compute the weighted average (using as weight each frame's duration in sections), the weighted standard deviation, minimum and maximum value for each feature, both on the output and target piano rolls. We use as features the weighted average, the weighted standard deviation, minimum and maximum computed on each consonance measures on the output piano roll.

3.11 Polyphony level

We assume that a mistake is more salient when it is the only note being played. Conversely, if a big chord is supposed to be played, but few notes are detected, this will be noticeable.

We compute the difference in polyphony level as a time series:

$$\text{Poly}(t) = \left| \sum_{0 \leq p < 88} \hat{M}[p,t] - \sum_{0 \leq p < 88} M[p,t] \right| \tag{21}$$

We then use as features the mean, standard deviation, minimum and maximum of this series.

4 Summary

We provide a table summarising all the features. The first column corresponds to feature groups (as described in the sections above), the second column describes each scalar value that can be found within that feature group, and the last column describes whether higher is better for that metrics: "Yes" if higher is better, "No" if lower is better, "/" when it depends on other factors.

Feature group	Sub-feature	Higher is better?
Benchmark Framewise metrics	Precision	Yes
	Recall	Yes
	F-measure	Yes
Benchmark Onset-only notewise metrics	Precision	Yes
	Recall	Yes
	F-measure	Yes
Benchmark Onset-Offset notewise metrics	Precision	Yes
	Recall	Yes
	F-measure	Yes
Framewise mistakes in highest voice	Precision	Yes
	Recall	Yes
	F-measure	Yes
Framewise mistakes in lowest voice	Precision	Yes
	Recall	Yes
	F-measure	Yes
Notewise mistakes in highest voice	Precision	Yes
	Recall	Yes
	F-measure	Yes
Notewise mistakes in lowest voice	Precision	Yes
	Recall	Yes
	F-measure	Yes
Loudness	Normalised false negative loudness	No
	False negatives loudness ratio	No
Binary out-of-key false positives	Proportion among false positives	No
	Proportion among detected notes	No
Non-binary out-of-key false positives	Average key-disagreement of false positives	No
	Average key-disagreement of false positives normalised by the average key-disagreement of all detected notes	No
Framewise semitone errors	Proportion among false positives	/
	Proportion among detected notes	/
Framewise octave errors	Proportion among false positives	/
	Proportion among detected notes	/
Framewise third-harmonic errors	Proportion among false positives	/
	Proportion among detected notes	/

Feature group	Sub-feature	Higher is better?
Notewise semitone errors	Proportion among false positives	/
	Proportion among detected notes	/
Notewise octave errors	Proportion among false positives	/
	Proportion among detected notes	/
Notewise third-harmonic errors	Proportion among false positives	/
	Proportion among detected notes	/
Repeated notes	Proportion among false positives	/
	Proportion among detected notes	/
Merged notes	Proportion among false positives	/
	Proportion among detected notes	/
Rhythm histogram spectral flatness	Value computed on output	/
	Relative difference between value computed on output and on target	/
Rhythm dispersion	Mean centre drift	No
	Minimum centre drift	No
	Maximum centre drift	No
	Mean cluster standard deviation difference	/
	Minimum cluster standard deviation difference	/
	Maximum cluster standard deviation difference	/
Consonance measures	Mean of `hutch_78_roughness`	/
	Standard deviation of `hutch_78_roughness`	/
	Minimum of `hutch_78_roughness`	/
	Maximum of `hutch_78_roughness`	/
	Mean of `har_18_harmonicity`	/
	Standard deviation of `har_18_harmonicity`	/
	Minimum of `har_18_harmonicity`	/
	Maximum of `har_18_harmonicity`	/
	Mean of `har_19_corpus`	/

Feature group	Sub-feature	Higher is better?
	Standard deviation of `har_19_corpus`	/
	Minimum of `har_19_corpus`	/
	Maximum of `har_19_corpus`	/

Table 2: Summary of all the proposed evaluation metrics.

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)*, 2009.

[2] Tian Cheng. *Exploiting Piano Acoustics in Automatic Transcription*. PhD thesis, Queen Mary University of London, 2016.

[3] Peter Harrison and Marcus T. Pearce. Simultaneous consonance in music perception and composition. *Psychological Review*, 127(2):216, 2020.

[4] James D. Johnston. Transform coding of audio signals using perceptual noise criteria. *IEEE Journal on selected areas in communications*, 6(2):314–323, 1988.

[5] Kevin P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

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

[7] Adrien Ycart, Lele Liu, Emmanouil Benetos, and Marcus T. Pearce. Investigating the perceptual validity of evaluation metrics for automatic piano music transcription. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, Accepted, 2020.