nature machine intelligence

Supplementary information

https://doi.org/10.1038/s42256-026-01279-9

Machine learning of artistic fingerprints in jazz

In the format provided by the authors and unedited

Contents

S.1 Supplementary Methods and Analyses **2**
S.1.1 Jazz Scale Estimation with `aeberscale` 2
S.1.1.1 Algorithm 2
S.1.1.2 Evaluation 3
S.1.1.3 Effect of Training Using Diatonic Scale Steps 4
S.1.2 Additional Idiomatic Melodic Patterns and Chord Voicings . . . 4
S.1.3 Principal Component Analysis Methods 5
S.1.4 Validation of Data Splits 5
S.1.4.1 Composition-Level Entity Resolution 6
S.1.5 Validation of Automated Algorithm for Separating Melody . . . 7
S.1.6 Convolutional Neural Network Architectures 8
S.1.7 Data Augmentation Pipeline 9
S.1.8 Generating Multiple Input Representations 9

S.2 Supplementary Figures **11**

S.3 Supplementary Tables **37**

S.1 Supplementary Methods and Analyses

S.1.1 Jazz Scale Estimation with `aeberscale`

In the "Bag of features approach" section, we describe how chromatic scale steps are used to represent interval patterns and chord voicings. An alternative method of representing the same information is to use diatonic scale steps. When expressed in this way, the chromatic interval patterns $(0, 2, 4, 6)$, $(0, 2, 4, 5)$, and $(0, 2, 3, 5)$ are all expressed as the diatonic pattern $(0, 1, 2, 3)$.

In order to "diatonicise" the chromatic interval patterns contained within our dataset, we first need a mechanism for mapping each onto their diatonic equivalent. This requires some approximation of the underlying scale. While the concept of musical scale estimation has been addressed frequently before, the vast majority of existing algorithms consider only classical music, and in particular only the diatonic major and minor scale [e.g., 1]. Jazz, on the other hand, uses a much larger number of distinct scales, many of which cannot are not 7-note heptatonic scales. These include, for example, pentatonic, hexatonic (e.g., "whole-tone" scales), and octatonic (e.g., diminished and "bebop" scales). Existing key-finding algorithms are thus unlikely to be appropriate for jazz.

S.1.1.1 Algorithm

To this end, we introduce `aeberscale`, a simple algorithm for musical key estimation in jazz. As the name suggests, `aeberscale` is based on the "Scale Syllabus" developed by jazz educator Jamey Aebersold. The syllabus contains 25 distinct scales, grouped hierarchically ("Major", "Dominant 7th", "Minor", "Half-Diminished", and "Diminished"). Aebersold's syllabus is well-known to musicians, both for its use in educational contexts and for its inclusion on the back page of modern copies of the "Charlie Parker Omnibook" [here, see 2].

The algorithm component of `aeberscale` is an adaptation of the Krumhansl-Schmuckler key-finding algorithm [described in 1]. For each scale and root note, we generate a "key profile": a binary vector of 12 values representing the presence or absence of the 12 pitch classes relative to that scale and root. Given a piece of music, we compute an equivalent "input vector": a continuous-valued vector with each of the 12 values representing the total duration of that pitch class in the piece. To estimate the scale, we then compute the correlation coefficient between the input vector and all key profiles; the profile that yields the highest correlation gives the preferred key.

In cases of rotational equivalence — where the same pitch classes may be contained in multiple "scales" (seen, for instance, in the whole-tone scale) — we adopt a simple heuristic, selecting the root based on the note with the longest duration in the piece. Future work may attempt to address this limitation by conducting perceptual experiments with participants familiar with jazz, in order to generate continuous-valued (as opposed to binary) key-profiles.

When computing the key of an input, we divide this into non-overlapping windows containing five seconds of music. We select this value as it is the

approximate duration of four bars at the average musical tempo annotated in JTD [194 quarter-note beats-per-minute: see 3]: four-bar phrases are extremely common in jazz [4]. In cases where fewer than 64 notes are found in this range, we append notes played before the earliest onset, in cases where more than 64 notes are found, we omit the last notes played. Again, we select 64 notes here as it is the average number of notes contained within a typical four bar passage in JTD.

S.1.1.2 Evaluation

In order to evaluate the effectiveness of `aeberscale`, we need a dataset of jazz improvisations annotated with the scales contained in the Aebersold syllabus. Luckily, later editions of the syllabus include a demonstration CD, featuring acclaimed soprano saxophonist David Liebman improvising (in the left audio channel) along each scale, with appropriate accompaniment on piano from Jamey Aebersold (in the right audio channel). While Liebman's solos do mainly stick to the target scale, out-of-scale and non-diatonic notes feature frequently (especially so in Aebersold's accompaniment), making for a relatively naturalistic set of stimuli to test our algorithm against.

We split the left and right audio channels of every recording, then process the left channel (Liebman's saxophone) with `basic-pitch` [5] and the right channel (Aebersold's piano) with the same model used to transcribe both JTD and PiJAMA [6]. This produces two MIDI files, which we combine together and preprocess with the pipeline described in the "Bag of features approach" section (e.g., by quantising notes). Excluding recordings which either (a) consist solely of "tuning notes" or warm-up exercises, or (b) feature multiple scales from the syllabus, we analyse a total of 22 recordings with a combined duration of 56 minutes and which contain 35,731 MIDI notes.

We divide up each recording into non-overlapping windows (see above) and estimate the scale contained within each window using the `aeberscale` algorithm. This produces a set of ordered pitch classes A, which we compare to the ground truth pitch class set for the recording B. As $|A|$ may not be equal to $|B|$, we express the similarity between A and B using the Jaccard index $J(A, B)$, calculated as:

$$J(A, B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{S.1}$$

where $0 \leq J(A, B) \leq 1$.

As a naive baseline, for every pitch class set A we also compute a random set R by drawing a random scale and root note from the syllabus. We then compute the index $J(R, B)$.

Averaged across all recordings, the mean $J(A, B) = 0.821$ (SD = 0.212), whereas the baseline $J(R, B) = 0.434$ (SD = 0.189). This indicates that, despite its simplicity, the `aeberscale` algorithm is able to efficiently capture the types of scales used in polyphonic jazz improvisation. We release `aeberscale` as a

separate Python package to enable other researchers to take advantage of and build upon this work in the future (see the "Code availability" section).

S.1.1.3 Effect of Training Using Diatonic Scale Steps

We apply the `aeberscale` algorithm to the main dataset described in this paper in order to transform chromatic interval patterns into diatonic ones. We refer to this as feature type (b) in Extended Data Table 1. In total, 2,070 valid diatonic scale step patterns are extracted from our dataset, and 478 valid diatonic chord voicings. We train each model described in the "Bag of features approach" section on this dataset and find that they perform approximately 2.5 times worse than when using chromatic interval patterns. This might suggest that chromatic interval patterns are better at distinguishing between the pianists in our dataset than diatonic ones.

To explore this possibility further, we additionally control for mode preference, including the raw mode counts (e.g., major, melodic minor) alongside the matrix of harmonic and melodic feature counts. We refer to this as feature type (c) in Extended Data Table 1. We find that models trained on this dataset perform slightly better than those trained on diatonic scale steps alone i.e., feature type (b). This may suggest that it is beneficial to learn general characteristics around scale preference — for instance, Thelonious Monk's noted penchant for whole-tone scales [7] — alongside both melodic patterns and chord voicings.

S.1.2 Additional Idiomatic Melodic Patterns and Chord Voicings

Here, we offer additional discussion of several idiomatic melodic patterns and chord voicings featured in Supplementary Figures S.5–S.42.

Tommy Flanagan (Supplementary Figure S.22), perhaps best known for accompanying saxophonist John Coltrane on his album "Giant Steps", is strongly associated with a $(0, 2, 4, 7)$ scalar pattern (Supplementary Figures S.47–S.49) that Coltrane used throughout this record, Frieler *et al.* [discussed also in 8]. Hank Jones is associated with the alternating $(0, -2, 0, -2)$ pattern, used in various sequential patterns (Supplementary Figures S.50–S.51) and as a melodic embellishment (Supplementary Figures S.52–S.53). McCoy Tyner (Supplementary Figure S.18) is associated with multiple melodic patterns that outline a perfect fourth interval (± 5), including $(0, -5, -7, 0, -5)$ and $(0, 5, 3, 0)$: Tyner himself has acknowledged that the use of this interval is "one of the characteristics of [his] style" [9, p. 97]. Thelonious Monk (Supplementary Figure S.21) is associated with a descending whole-tone pattern $(0, -2, -4, -6, -8)$ that theorist Gunther Schuller [7, p. 231] claimed Monk was "addicted" to.

Supplementary Figures S.23–S.42 contain analogous results for the chord voicings. While we reserve a full analysis for subsequent work, several observations can nonetheless be made here. For instance, Bill Evans (Supplementary Figure S.25) is strongly associated with the cluster $(0, 1, 5)$, which can be considered a minor voicing containing the ninth, minor third, and fifth; it was in

the use of such "rootless" voicings that Evans was particularly known for [10]. Many other voicings can be considered elaborations of an underlying dominant harmony, such as the $(0, 16, 22, 27)$ voicing associated with Cedar Walton (Supplementary Figure S.8) — commonly known to rock musicians as the "Jimi Hendrix chord" [11]. Finally, McCoy Tyner (Supplementary Figure S.37) is again associated with voicings built on the fourth interval, including $(0, 5, 10)$ and $(0, 6, 11)$.

S.1.3 Principal Component Analysis Methods

Here, we describe the process used to perform principal component analysis (PCA) in the "Bag of features approach" section.

First, we process the raw feature counts obtained in the "Bag of features approach" section with TF-IDF, centre each on their mean value (i.e., $z = \frac{x^{'} - \mu}{\sigma}$, where $x^{'}$ is the value for feature x after TF-IDF), and scale the results between $(0, 1)$, according to the minimum and maximum values for each feature. We then apply PCA to reduce this high-dimensional feature space into a smaller set of axes (or components), each of which captures a single, distinct dimension of variation in melodic language. We then compute the "loading" of every feature onto each component: this indicates how strongly the feature contributes to the component, with the sign indicating which pole of the axis it belongs to. Finally, we project each performer in our dataset onto the same space by taking the dot product of their averaged feature counts with every component. In this case, performers who frequently use positively-loading features will themselves score positively on that component, and vice versa (Figure 3).

Analysis of principal components 1 and 2 is contained in the "Bag of features approach" section. Melodic patterns that load positively onto component 3 are scalar fragments that approach a perfect fourth, including $(0, 1, 3, 5)$ and $(0, 2, 4, 5)$, while patterns that load negatively instead outline an augmented fourth, such as $(0, 3, 6, 3)$. Several patterns that load positively onto component 4 outline ascending major $(0, 4, 7, 11)$ and minor $(0, 3, 7, 10)$ seventh arpeggios. Perhaps unsurprisingly given the patterns in Figure 2a (alongside the transcriptions in Supplementary Figures S.43–S.44), Bill Evans loads positively onto this component; interestingly, so also do McCoy Tyner and Oscar Peterson.

Finally, we note that the lower-dimensional space obtained through PCA appears to embody a degree of invariance to melodic inversion, with patterns that outline the same intervallic patterns such as $(0, -5, 0, -5)$ and $(0, 5, 0, 5)$ typically mapped to similar coordinates regardless of their melodic contour. There may be some invariance to mode, too, with the major seventh $(0, 4, 7, 11)$ and minor $(0, 3, 7, 10)$ arpeggios mapping onto component 4 with a similar loading.

S.1.4 Validation of Data Splits

In the "Dataset" section, we describe how we split the recordings in our dataset into training, validation, and testing subsets by stratifying according to the type

of ensemble featured (either trio or unaccompanied). However, this could introduce potential confounds and data leakage, as models trained on these splits may overfit to performer-specific repertoire choices, recording-era characteristics, or session-specific regularities. These tendencies have been described as either the "album" or "composition" effect: in the first case, a model memorises acoustic information contained in multiple recordings from the same session, or by the same producer [12]; in the second, a model memorises consistent melodic patterns or chord progressions that appear within multiple interpretations of the same composition [13].

To test this hypothesis, we create two additional sets of data splits, stratified in such a way to prevent data leakage between recordings (a) from the same album or (b) of the same composition (here, see Supplementary Information, Section S.1.4.1). We keep the size of these splits identical to those discussed in the "Data splits" section, and also stratify them by ensemble type. We then train the best-performing models from the "Bag of features approach" section (Logistic Regression) and "Representation learning approach" section (ResNet-50, with data augmentation) on these splits and compute track-level accuracy scores against the validation data. Note that the multi-input model described in the "Multiple input representations approach" section is not tested, due to the longer training time.

Next, we create an additional set of 20 data subsets by splitting the full dataset randomly (as in the "Data splits" section). As before, we train the same model architectures on each subset and compute the validation accuracy. We then perform a one-sample, left-tailed t-test between the accuracy measurements obtained when stratifying splits and the equivalent null distribution obtained from random splitting, in order to determine whether model performance decreases significantly when stratifying. We find no evidence for such a decrease: when stratifying by album, $t(19) = -1.475,\ p = .08$ for the LR, $t(19) = -0.984,\ p = .17$ for the ResNet; when stratifying by composition, $t(19) = 0.567,\ p = .71$ for the LR, $t(19) = 0.764,\ p = .77$ for the ResNet.

This suggests that neither the "album" or "composition" effects significantly impact our models when training them on random splits of the data. One potential explanation could be that our use of MIDI and our pre-processing of the velocity information contained within it (see the "Input representation" section) strips most of the acoustic information from a signal [here, see 14, Table 4]. Another could be that performances of the same jazz composition usually differ significantly from each other, due to the emphasis on improvisation.

S.1.4.1 Composition-Level Entity Resolution

Before carrying out these experiments, we first needed to perform composition-level entity resolution for the recordings in our dataset. This has not previously been accomplished for either JTD or PiJAMA and is, in fact, a non-trivial problem, because the title of a jazz recording often does not always denote the composition being performed. In some cases, the differences between titles may be minor (e.g., "A Night in Tunisia [Remastered]" and "Night in Tunisia

[Alternative Take]"). In other cases, domain expertise is required to select the correct match, such as for the small number of jazz standards that have come to be known by several titles (e.g., "Corcovado" and "Quiet Nights of Quiet Stars").

We designed a "human-in-the-loop" system to perform this task. First, the title of a recording was pre-processed with a range of regular expressions to remove common phrases, such as "Alternate Take", "Live", and "Remastered". The title was then passed into a pre-trained sentence embedding model [15], and the cosine similarity S_C was computed between all pairwise combinations of embeddings to produce a square matrix. We identified clusters in this matrix where the similarity between two titles A and B was above a pre-defined threshold — which, after some manual tuning, we set to $S_C(A, B) \geq 0.9$. Track titles within each cluster were set to a single, unique name.

Next, we merged remaining titles together using a hand-curated list of alternative jazz titles. In the case of a "contrafact", where the same chord sequence is set to a different melody to create a new piece [4], the composition where the chord sequence originated from was treated as the canonical entity. This was because our dataset includes JTD, which only includes the "solo" section of a performance — thus, only the chord sequence would be expected to be used by the performers in these recordings. Finally, the first author (a jazz expert) checked the identified composition against the original recording title and corrected any mistakes.

A total of 914 unique compositions are performed across the $1,629$ recordings in our dataset. Not including contrafacts, some of the most frequently played compositions are "Like Someone in Love" (17 recordings), "My Romance" (15), and "Stella by Starlight" (14) — all of which are well-known jazz standards [16]. We release these annotations to the community (see the "Code availability" section), and anticipate that they may prove useful in a range of music information retrieval tasks, such as cover song or contrafact identification [see, e.g., 17].

S.1.5 Validation of Automated Algorithm for Separating Melody

In the "Bag of features approach" section, we describe an automated algorithm used to isolate monophonic melodic passages from accompaniment. To validate this algorithm, we manually annotate the melody within a subsection of clips from our dataset. Our procedure here is identical to the validation conducted by Norgaard *et al.* [18], who used a similar algorithm to separate melody from accompaniment in a different corpus of jazz piano solos.

We selected at random a single 30-second clip from the seven pianists with the greatest number of clips in our dataset (Brad Mehldau, Keith Jarrett, Oscar Peterson, Bill Evans, Chick Corea, Kenny Barron, and John Hicks: see Figure 6). In total, these clips contain 3,544 MIDI notes (mean 506.286 notes per clip, $SD = 175.743$). The first author, a jazz expert, isolated a monophonic melodic line from each clip, retaining a total 1,017 notes (mean 145.286 notes per clip,

$SD = 71.129$). Processing each clip with the algorithm retains 1,042 notes (mean 148.857 notes per clip, $SD = 60.880$).

We then calculate the error rate in the same way as Norgaard *et al.* [18]:

$$ER(A, B) = \frac{|A| - |A \cap B|}{|A|} \tag{S.2}$$

where A is the set of melodic notes in the clip annotated by the jazz expert, B is the set of melodic notes identified by the algorithm, and $A \cap B$ is the intersection of both sets.

For our dataset, we find a mean error rate of 11.879% ($SD = 7.585\%$), slightly lower than the equivalent value found by Norgaard *et al.* [18, 12.2%]. As the authors of this study also concluded, however, we do not believe that notes retained by the algorithm in error form repeated patterns as consistently as actually played patterns do. Additionally, the rejection of pitch patterns that occurred fewer than 10 times across the dataset guards against the possibility that notes retained in error formed patterns used as part of our main analysis.

In Extended Data Figure 5 and Supplementary Figure S.1, we show both the original MIDI transcription, the expert annotations, and the automatically extracted melody for two performances by Kenny Barron and John Hicks. We make the expert-annotated melodies available as part of our dataset (see the "Code availability" section).

S.1.6 Convolutional Neural Network Architectures

Here, we describe the architectures of the two convolutional neural networks evaluated in the "Representation learning approach" section. The first model is inspired by the success of convolutional recurrent neural networks (CRNNs) in prior performer and composer identification work [6, 14, 19]. The input is processed using eight convolutional layers (with average pooling after layers two, four, and six) and a bidirectional gated recurrent unit with two layers and a hidden dimension of 256. This is then followed by a global max pooling layer to yield a 512-dimensional feature vector that passes through a fully connected layer and softmax function to generate class probabilities. Dropout is used with 50% probability on the classification head, as in Kong *et al.* [20]. This model has a total of 14.9 million learnable parameters.

The second model consists of the classical "ResNet-50" architecture, initially developed for computer vision and subsequently used for classical composer identification by Kim *et al.* [21] and Foscarin *et al.* [22]. The input is processed using an initial convolutional and max pooling layer, before passing into four residual blocks comprising three, four, six, and three layers of convolutional and batch normalization modules, each with skip connections. Average pooling is used after the final block to generate a 2048 dimensional feature vector, which again passes through a fully connected layer (without dropout) and softmax to generate class probabilities. This model has a total of 23.6 million learnable parameters. Both models are implemented in the `PyTorch` (version `2.4.1`) Python library [23].

S.1.7 Data Augmentation Pipeline

Here, we describe the data augmentation pipeline used to train all deep-learning models described within the paper. This pipeline consists of pitch, time, and velocity augmentation applied jointly to a single input clip and is broadly inspired by the procedures in Liu *et al.* [24]. Note that augmentation is applied only to recordings from within the training split of our dataset (see the "Dataset" section).

Pitch augmentation involves shifting all note pitch values p_0 in the clip by the value s, such that $p_{shift} = p_0 + s$, where $s \sim \mathcal{U}(-S, S)$ semitones and $\mathcal{U}$ is a discrete uniform distribution of integers. The upper limit S that values of s can take is defined separately for each individual clip (with a maximum value of ± 6 semitones) so as to always satisfy the inequality $20 < p^k_{shift} < 109,\ k = 1, 2, \ldots, K$, where K is the total number of MIDI notes. This is to ensure that augmented pitch values lie within the range of the piano keyboard and have the same intervallic structure as the original input.

Time augmentation applies dilation by scaling all note onset x_0 and offset y_0 times in the clip by the constant t, where $t \sim \mathcal{U}(0.8, 1.2)$ and $\mathcal{U}$ is a continuous uniform distribution. The new onset and offset times are defined as $x_{shift} = x_0 \times t$ and $y_{shift} = y_0 \times t$, respectively. When $t > 1$, any notes where $x_{shift} > 30$ or $y_{shift} > 30$ (i.e., that now lie outside of the clip boundaries: see the "Dataset" section) are removed. Vice-versa, when $t < 1$, the right "edge" of the clip will potentially now include notes just outside the original clip, with their onset and offset times also scaled by t.

Velocity augmentation involves perturbing the velocity of every note v_0 by the random variable d such that the new values $v_{shift} = v_0 + d$, where $d \sim \mathcal{U}(-12, 12)$ and $\mathcal{U}$ is a discrete uniform distribution of integers. Note that, unlike both pitch and time augmentation, we sample a new value of d from $\mathcal{U}$ for every value of v_0. Additionally, values of v_{shift} are clipped to satisfy the inequality $0 < v_{shift} < 128$ to conform with MIDI specifications.

The final part of our pipeline involves randomly adjusting the boundaries that are used to segment a single recording into the 30-second clips used during training (see the "Dataset" section). We randomly adjust the hop between two successive clips from the same recording to between 15 and 30 seconds (i.e., between 0 and 50% overlap). Note that a constant hop of 30 seconds is maintained for the validation and test datasets.

S.1.8 Generating Multiple Input Representations

Here, we describe the process by which the individual inputs used to train the multi-input model described in the "Multiple input representations" approach are created.

Similar to how n-grams were extracted (see the "Bag of features approach" section), to generate the melody representation we apply the skyline algorithm to the transcription to generate a vector of note events M. We remove velocity information from each note $m \in M$ by converting the channel dimension to

binary, setting a value of 1 for when a note is played and 0 when it is not — such that $m_{velocity} \in \{0, 1\}$. We remove rhythmic information by setting the inter-onset interval for each note to a constant value dependent on the length of M. Thus,

$$m_{on}^{i+1} - m_{on}^{i} = \frac{T}{|M|},\ i \in \{1, 2, \ldots, |M|\}\ , \tag{S.3}$$

where T is the width of the original input (i.e., 3,000). Note that, when $i = 1$, $m_{on} = 0$ and, when $i = |M|$, $m_{off}^{|M|} = T$. We then adjust the onset and offset times between successive notes to remove any overlap and maintain the original dimensionality of the input, such that $m_{on}^{i+1} = m_{off}^{i}$.

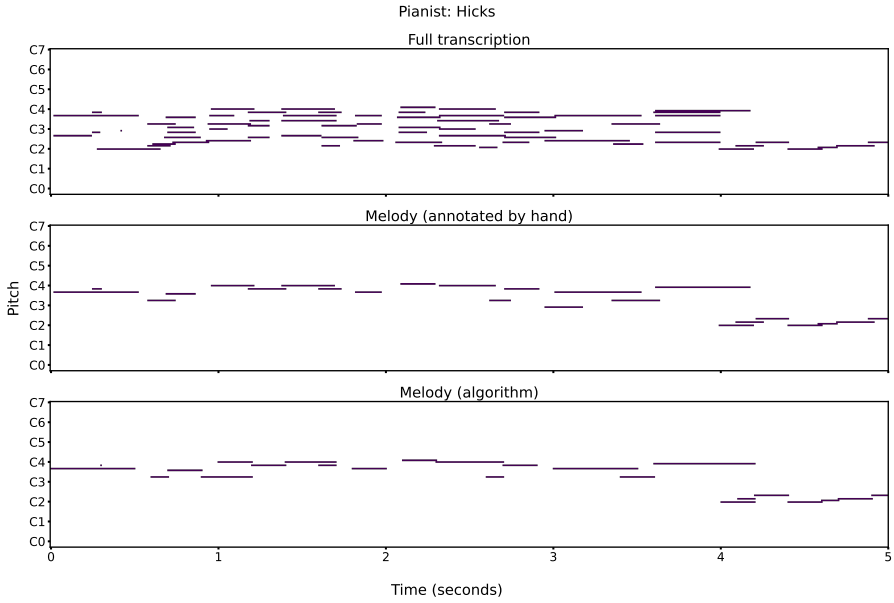
To generate the harmony representation, we quantise the transcription by grouping near-simultaneous notes into non-overlapping bins according to their onset time, with the size of each bin again set to 100 milliseconds as in the "Bag of features approach" section. This yields a vector of quantised bins H, where the size of every bin $|h| > 3$ for $h \in H$. Unlike the models trained in the "Bag of features approach" section, however, we do not need to set an upper limit to the number of notes in the chord. We set the inter-chord interval for all notes in each bin to a consistent value, adjust the onset and offset times to remove any overlap between successive chords, and convert the channel dimension to binary.

To generate the rhythm representation, we want to maintain only the onset and offset times for every note r in the original transcription. The influence of dynamics can be removed by converting the channel dimension to binary, as before. While it would be possible to remove the pitch dimension entirely as well, this could lead to situations where two notes with overlapping onset and offset times in the transcription are indistinguishable from each other in the derived representation. Instead, we randomise the pitch of each note r, such that $r_{pitch} \sim \mathcal{U}(21, 108)$, where $\mathcal{U}$ is a discrete uniform distribution of integers.

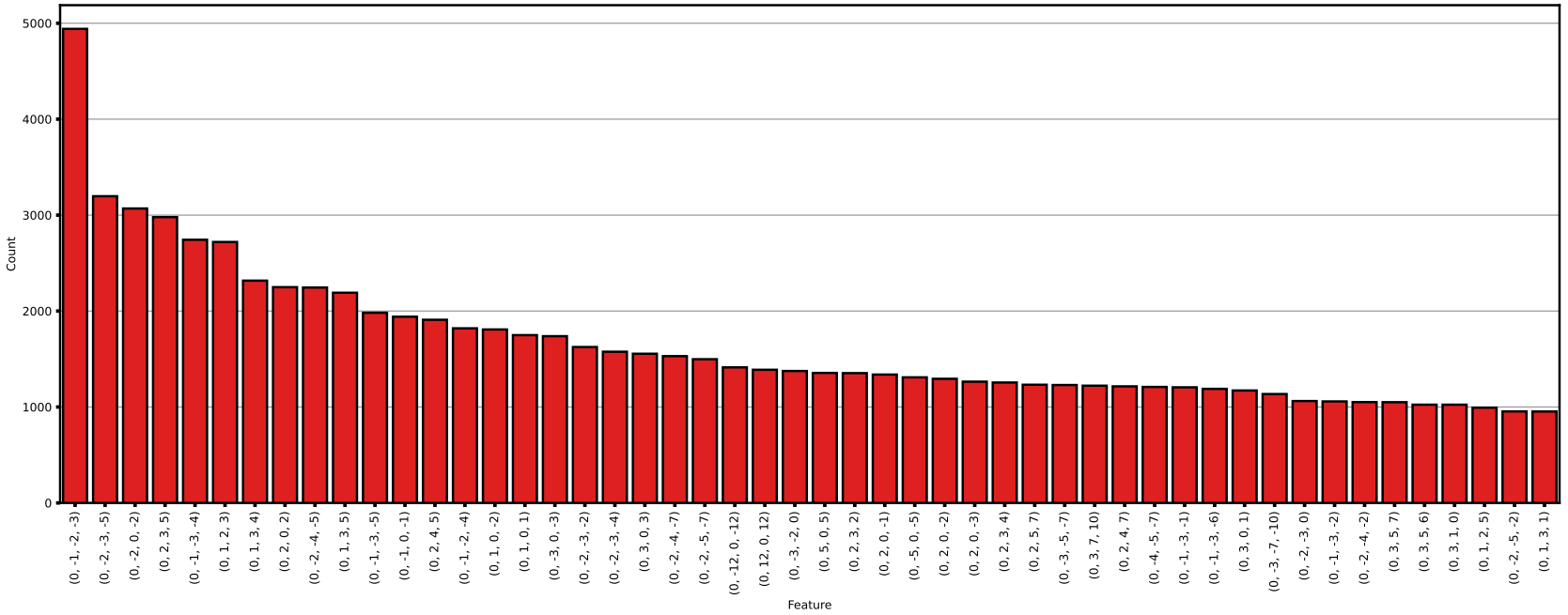
To generate the dynamics representation, we want to maintain only the velocity of each note in a transcription. We follow the process described to generate the harmony representation by binning near-simultaneous notes and adjusting their onset and offset times, but we do not set a lower boundary for the number of notes contained within a single bin (i.e., conceptually similar to the "snap to grid" function implemented in many commercial MIDI editors and digital audio workstations). Next, we randomise the pitch of each note in every bin. The representation derived from this process is thus the only one of the four used to train our model where the channel dimension of a note is a floating-point value, rather than an integer.

Compared with the models described in Supplementary Information, Section S.1.6, the best-performing multi-input model in Extended Data Table 3 has a total of 44.7 million learnable parameters.

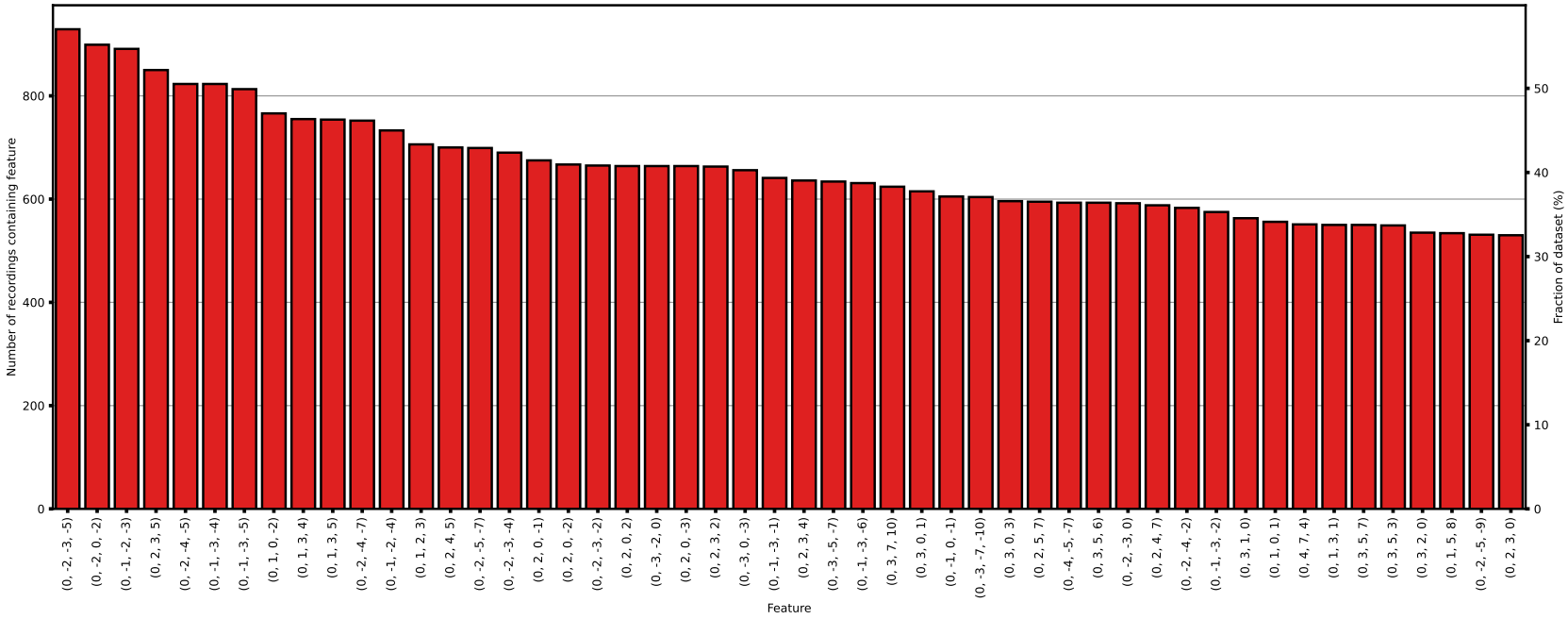
S.2 Supplementary Figures



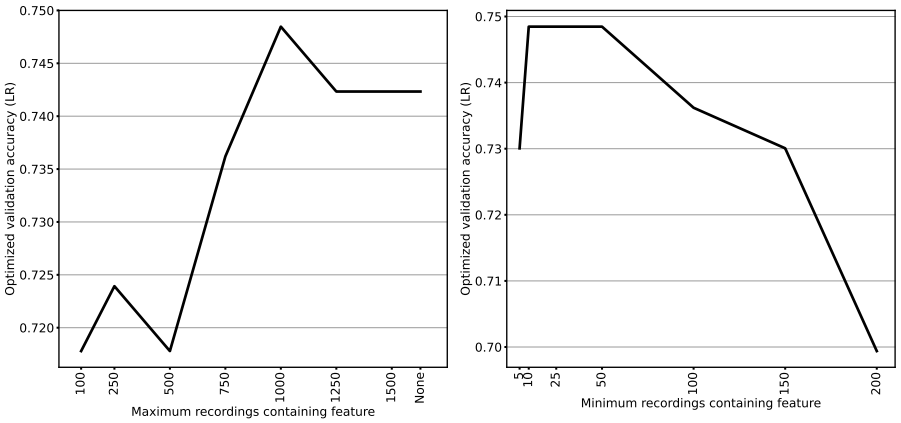
Supplementary Figure S.1: Melody extraction results. Panel layout as in Extended Data Figure 5, with the performance by John Hicks instead.



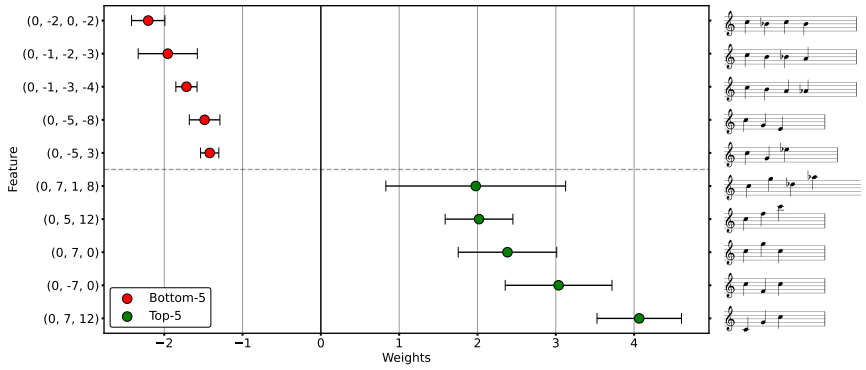
Supplementary Figure S.2: Most common melodic patterns, sorted by count. This figure shows the 50 most common 4-grams in the dataset, ranked by their total number of occurrences across all recordings (i.e., counts summed across recordings).



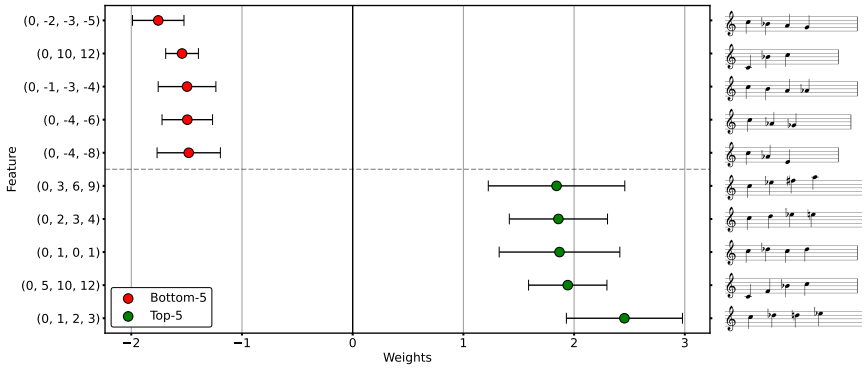
Supplementary Figure S.3: Most common melodic patterns, sorted by number of appearances. This figure shows the 50 4-grams that occur in the greatest number of distinct recordings (expressed as a proportion of the total size of the dataset), regardless of how many times they appear within each recording.



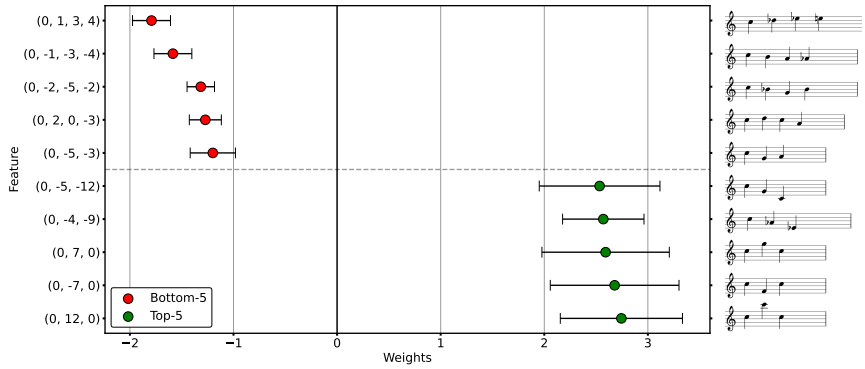
Supplementary Figure S.4: LR accuracy when discarding features contained in fewer or more than x recordings. The left panel shows changes in accuracy when features appearing in fewer than x unique recordings are discarded. The right panel shows equivalent changes when features appearing in more than x recordings are discarded. Datasets and optimisation procedures are identical to Extended Data Figure 6.



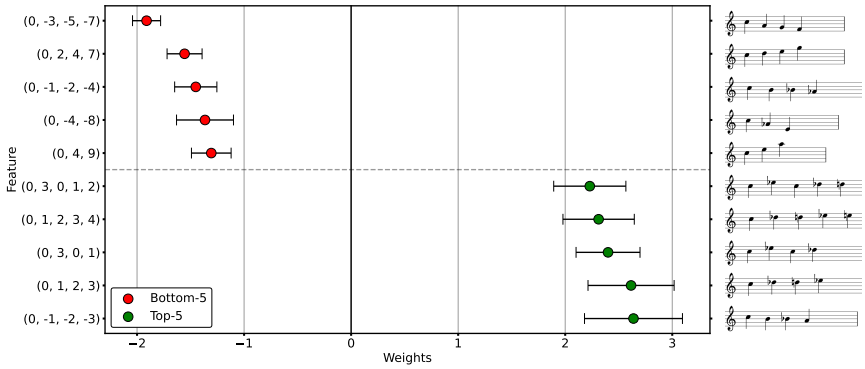
Supplementary Figure S.5: Predictive melody features, Abdullah Ibrahim. This figure (and those following) shows feature weights obtained for every performer, following Figure 2. Thus, data are presented as regression coefficients with error bars showing $\pm$ 1 SE, calculated by bootstrapping the dataset used to fit the model ($N = 1,000$ iterations).



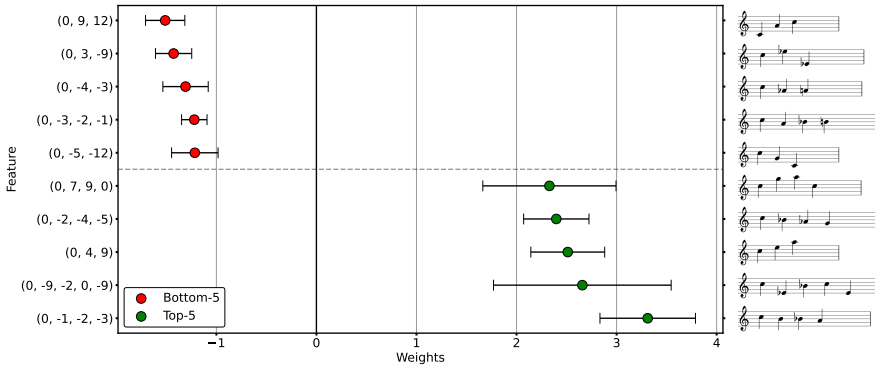
Supplementary Figure S.6: Predictive melody features, Ahmad Jamal.



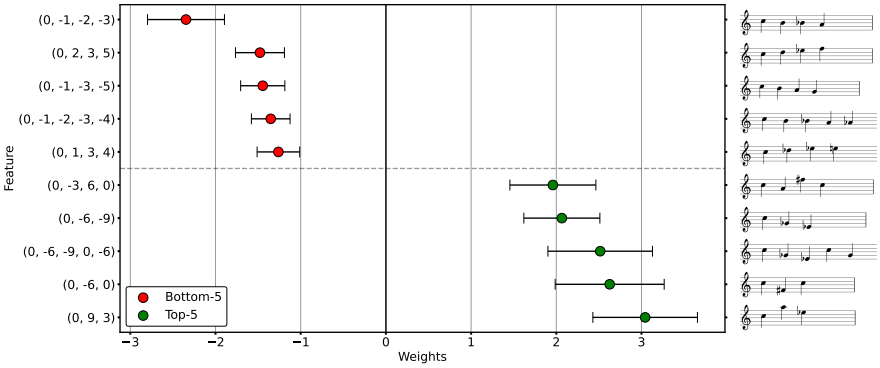
Supplementary Figure S.7: Predictive melody features, Brad Mehldau.



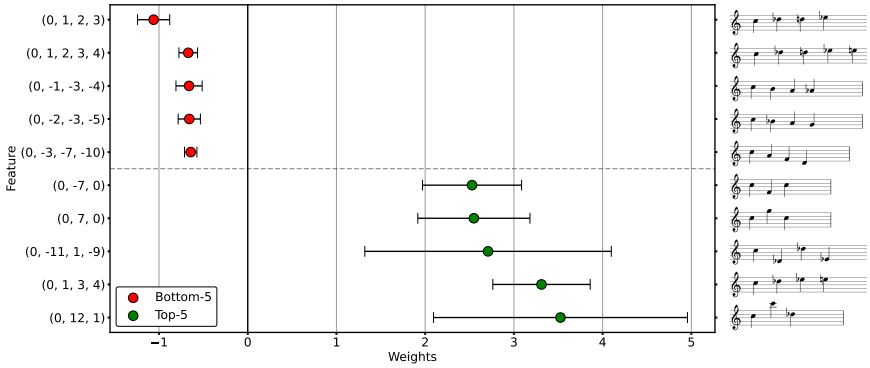
Supplementary Figure S.8: Predictive melody features, Cedar Walton.



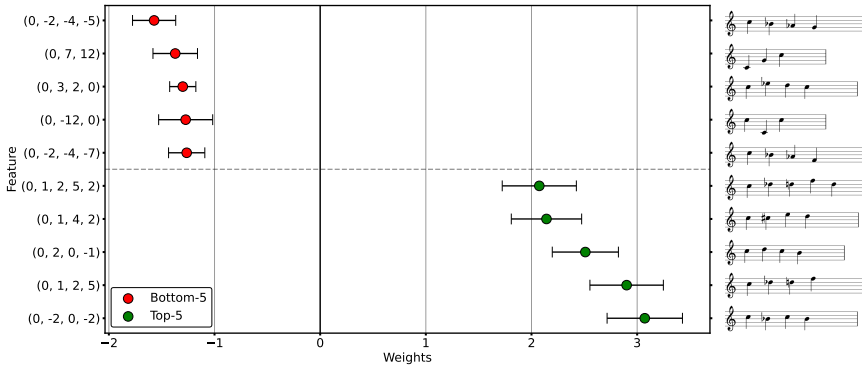
Supplementary Figure S.9: Predictive melody features, Chick Corea.



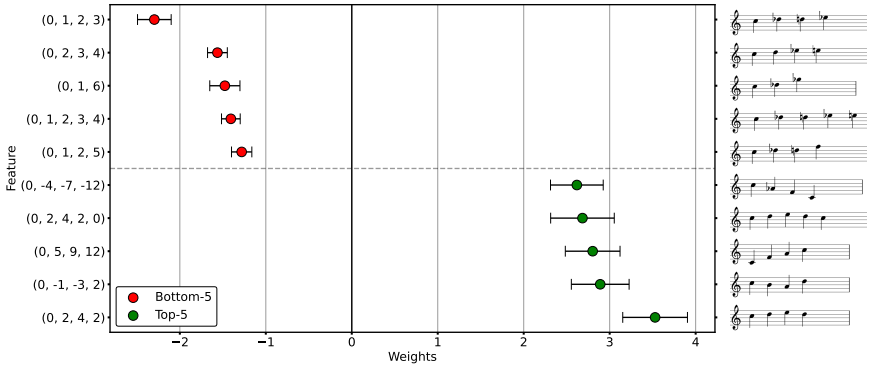
Supplementary Figure S.10: Predictive melody features, Gene Harris.



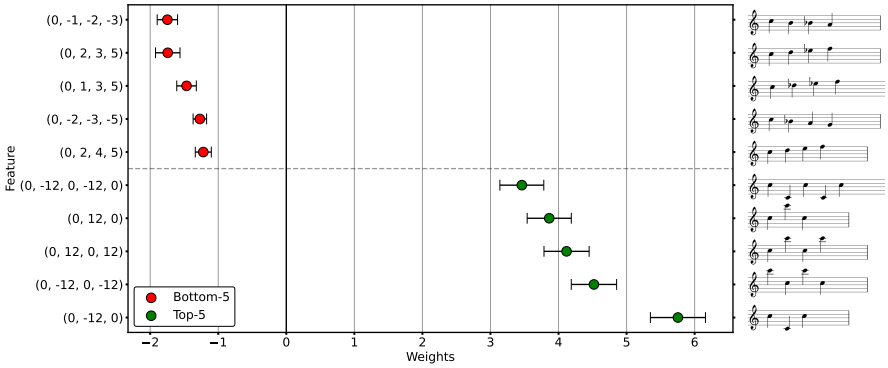
Supplementary Figure S.11: Predictive melody features, Geri Allen.



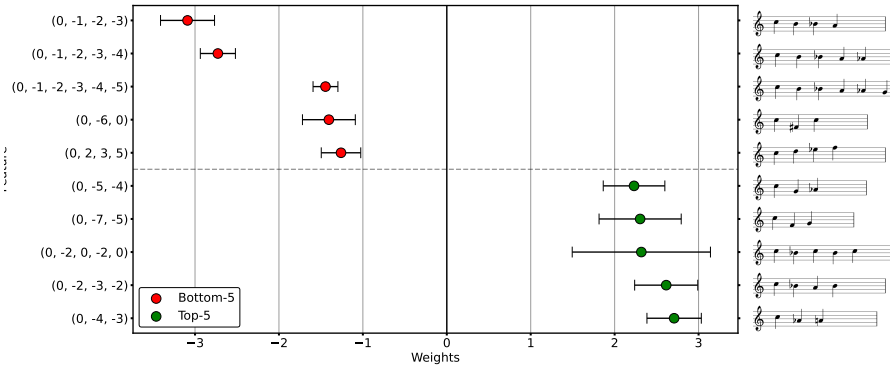
Supplementary Figure S.12: Predictive melody features, Hank Jones.



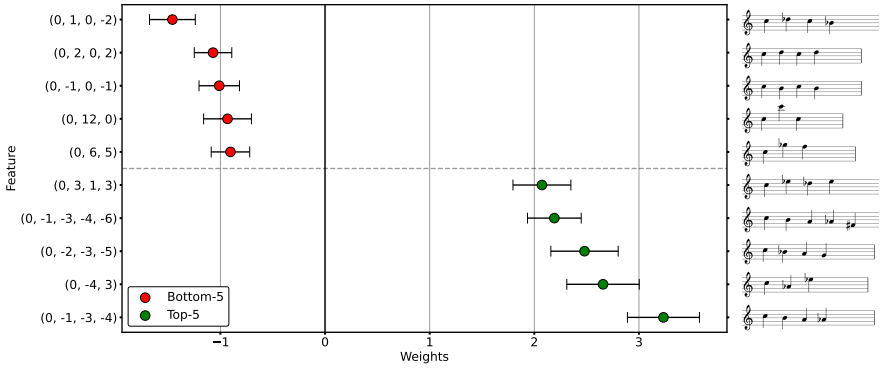
Supplementary Figure S.13: Predictive melody features, Junior Mance.



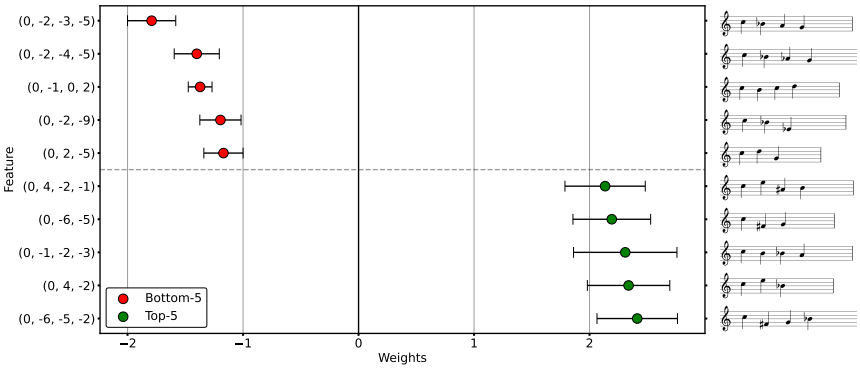
Supplementary Figure S.14: Predictive melody features, Junior Mance.



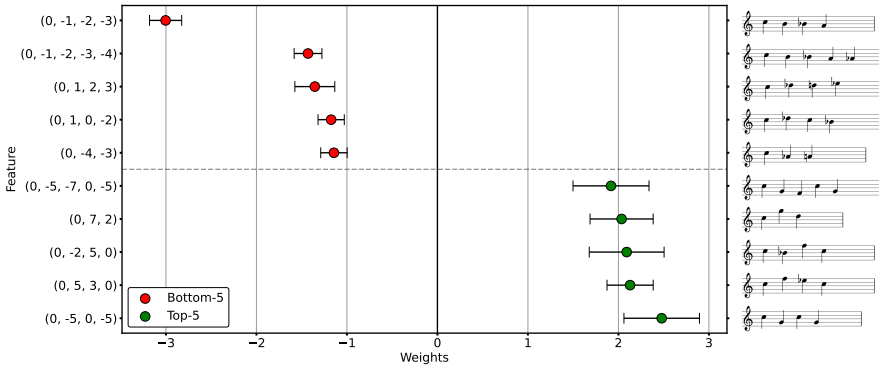
Supplementary Figure S.15: Predictive melody features, Keith Jarrett.



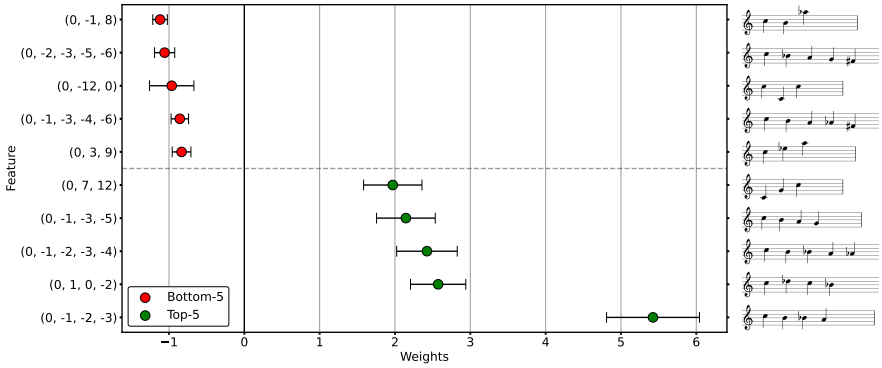
Supplementary Figure S.16: Predictive melody features, Kenny Barron.



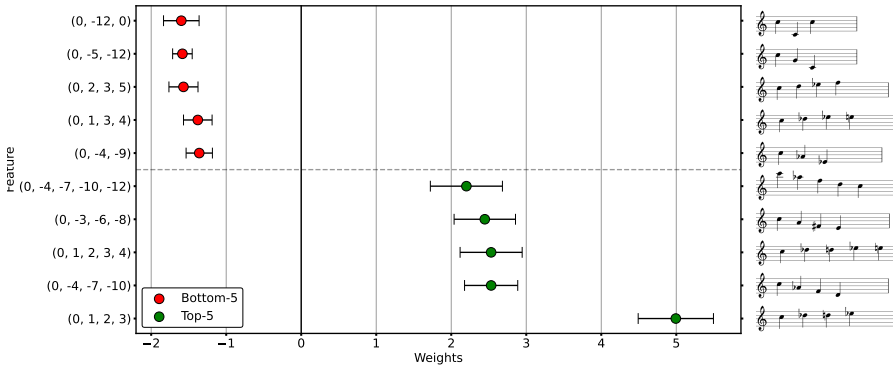
Supplementary Figure S.17: Predictive melody features, Kenny Drew.



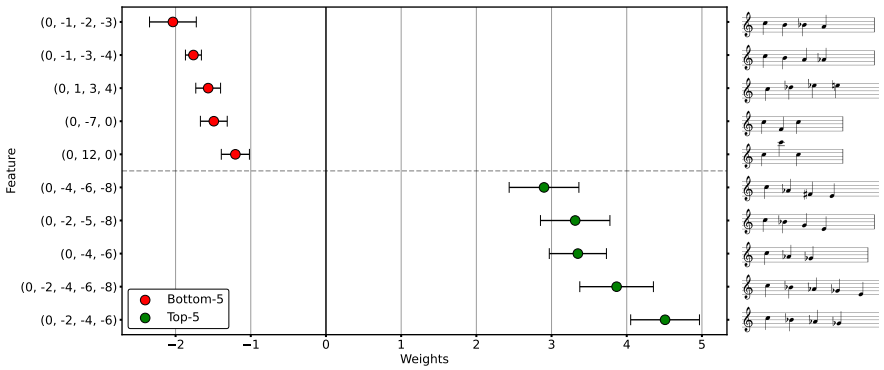
Supplementary Figure S.18: Predictive melody features, McCoy Tyner.



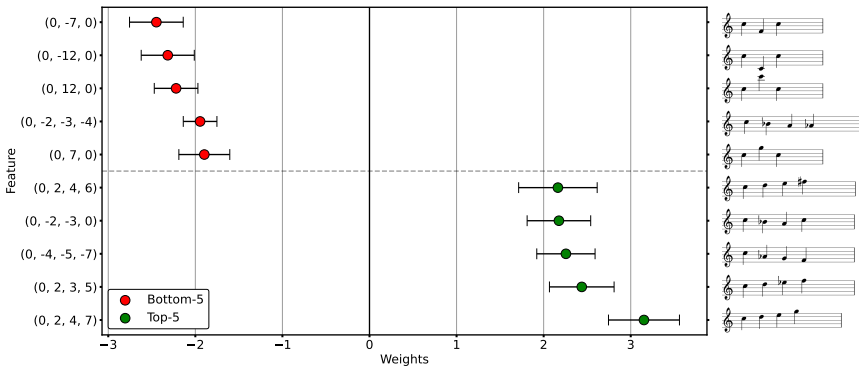
Supplementary Figure S.19: Predictive melody features, Stanley Cowell.



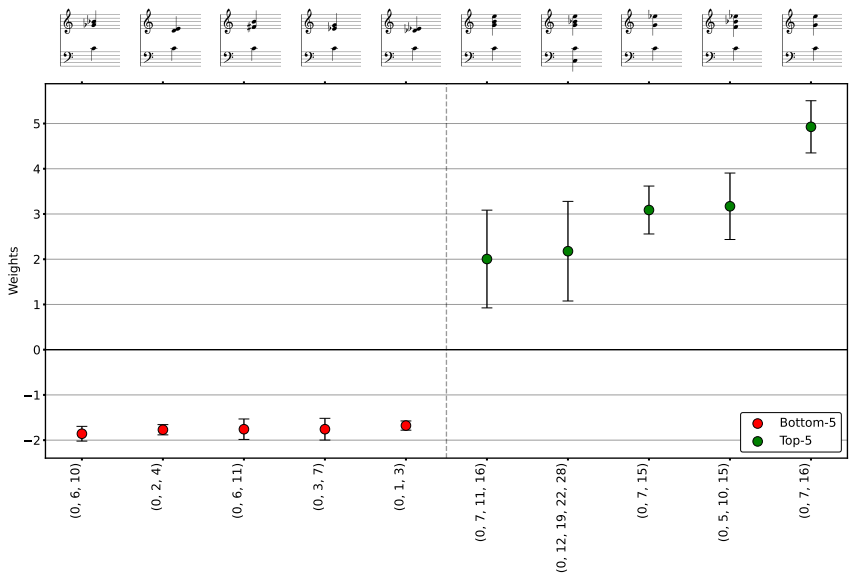
Supplementary Figure S.20: Predictive melody features, Teddy Wilson.



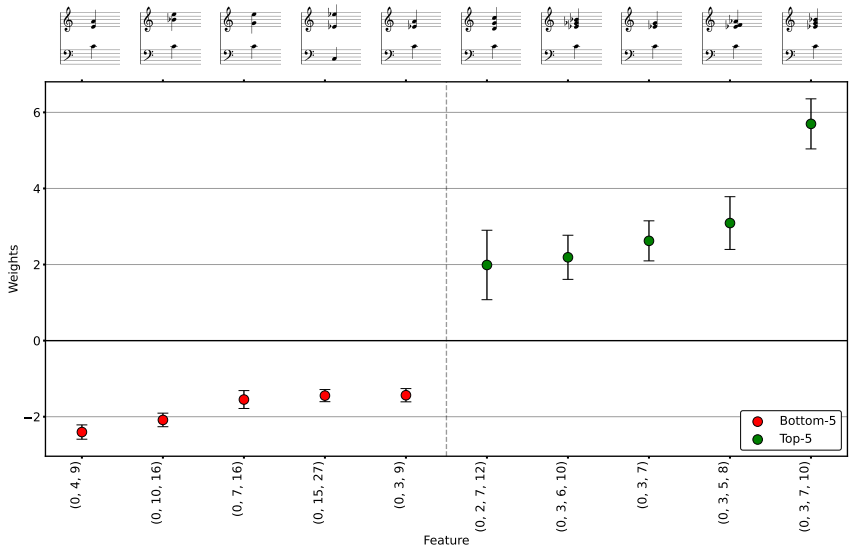
Supplementary Figure S.21: Predictive melody features, Thelonious Monk.



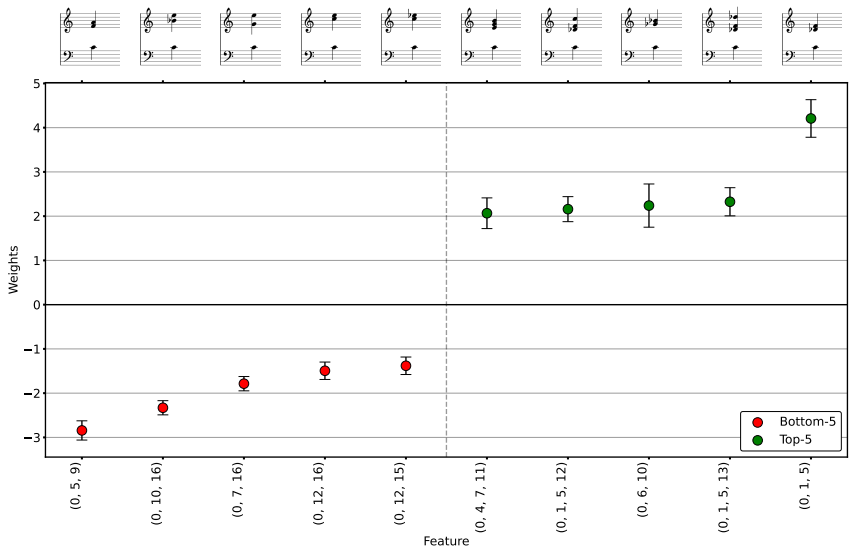
Supplementary Figure S.22: Predictive melody features, Tommy Flanagan.



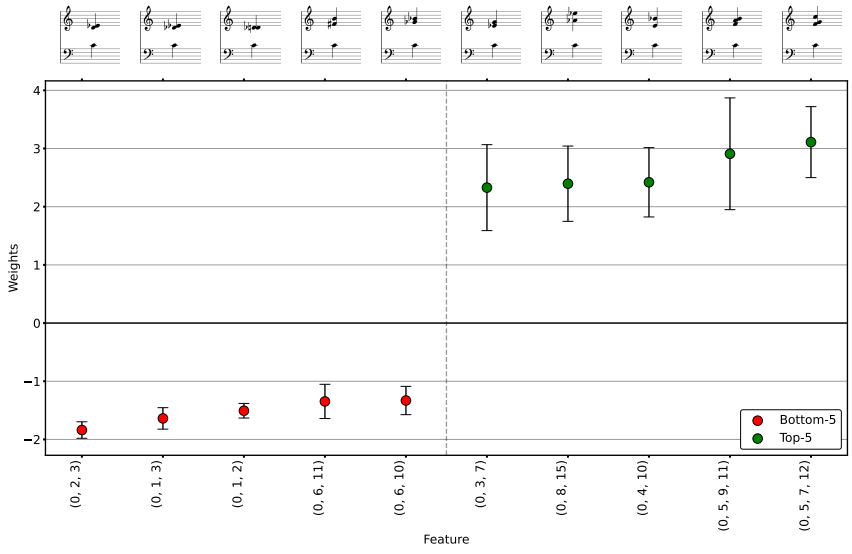
Supplementary Figure S.23: Predictive harmony features, Abdullah Ibrahim. Note that the orientation of the axis in these figures is reversed compared to the previous figures, in order to accommodate the "grand staff" notation.



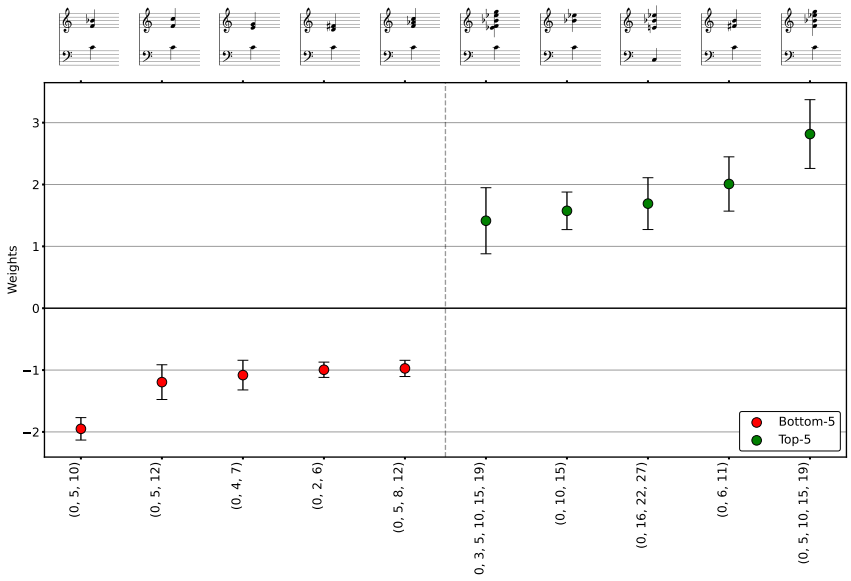
Supplementary Figure S.24: Predictive harmony features, Ahmad Jamal.



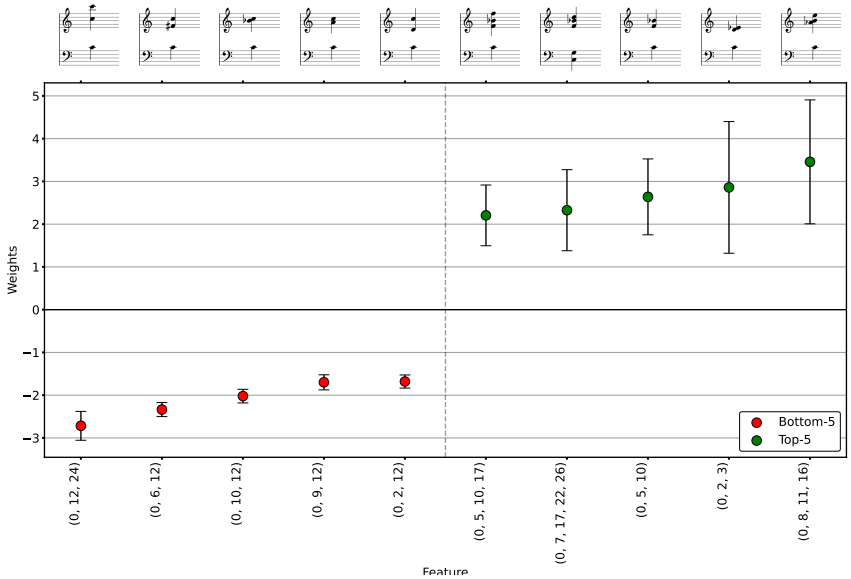
Supplementary Figure S.25: Predictive harmony features, Bill Evans.



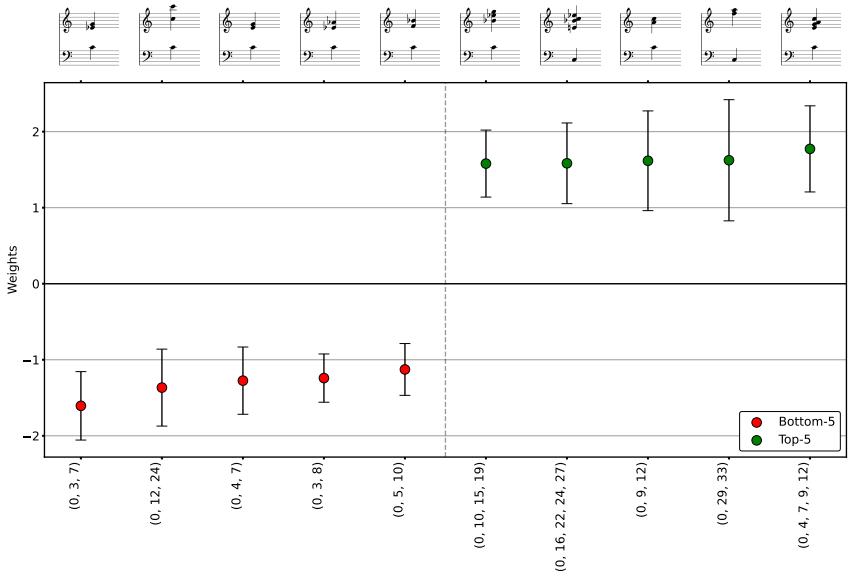
Supplementary Figure S.26: Predictive harmony features, Brad Mehldau.



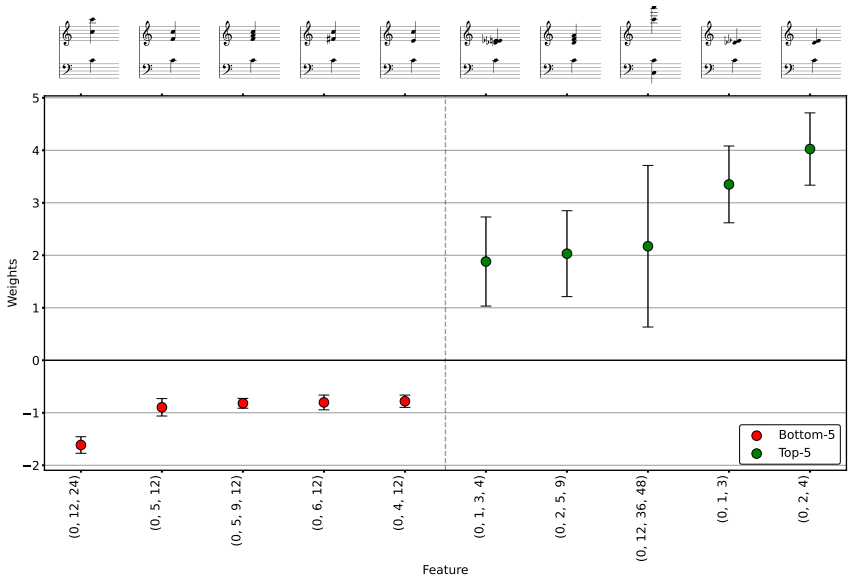
Supplementary Figure S.27: Predictive harmony features, Cedar Walton.



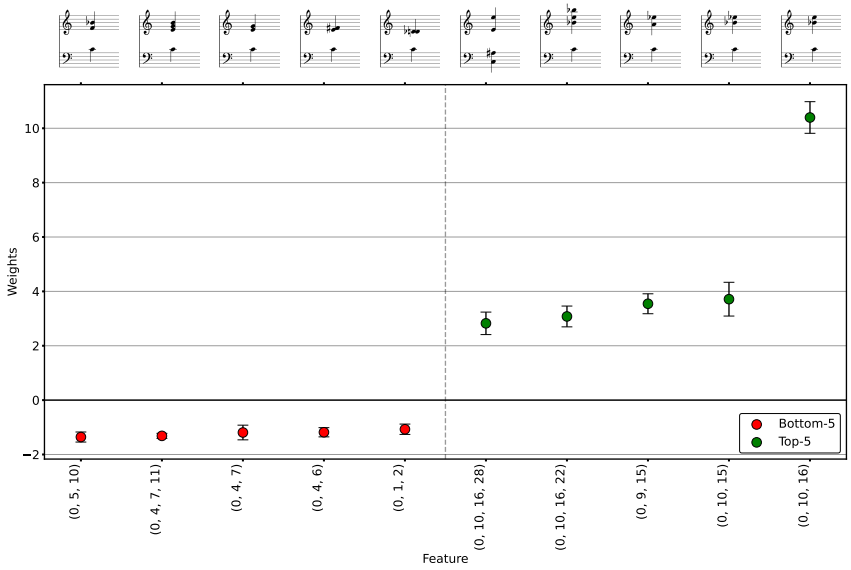
Supplementary Figure S.28: Predictive harmony features, Chick Corea.



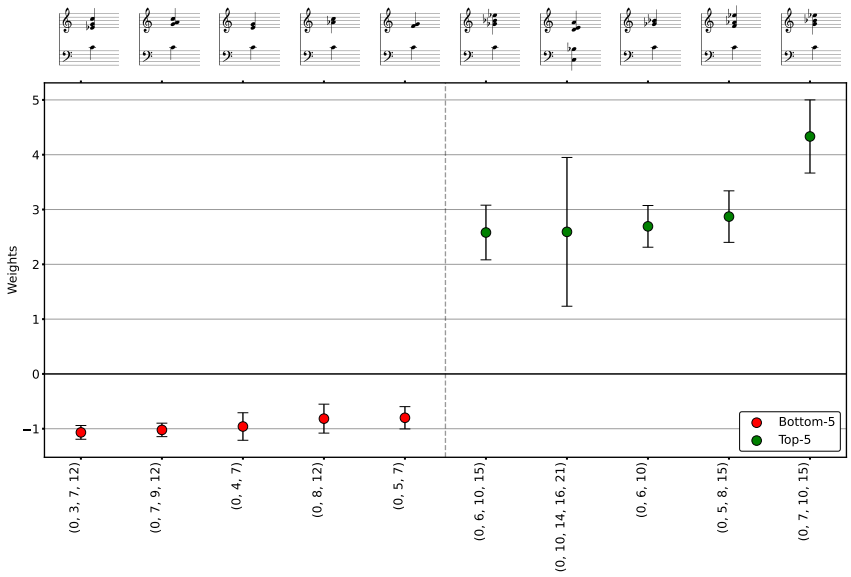
Supplementary Figure S.29: Predictive harmony features, Gene Harris.



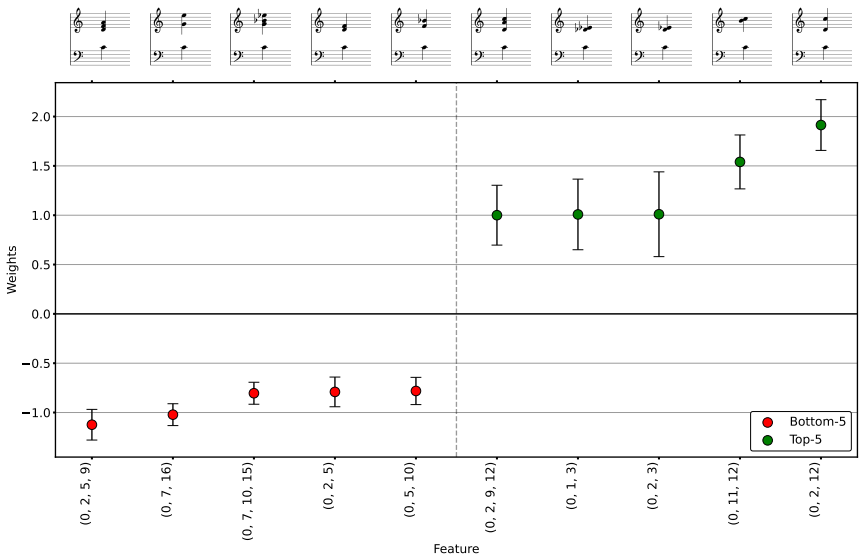
Supplementary Figure S.30: Predictive harmony features, Geri Allen.



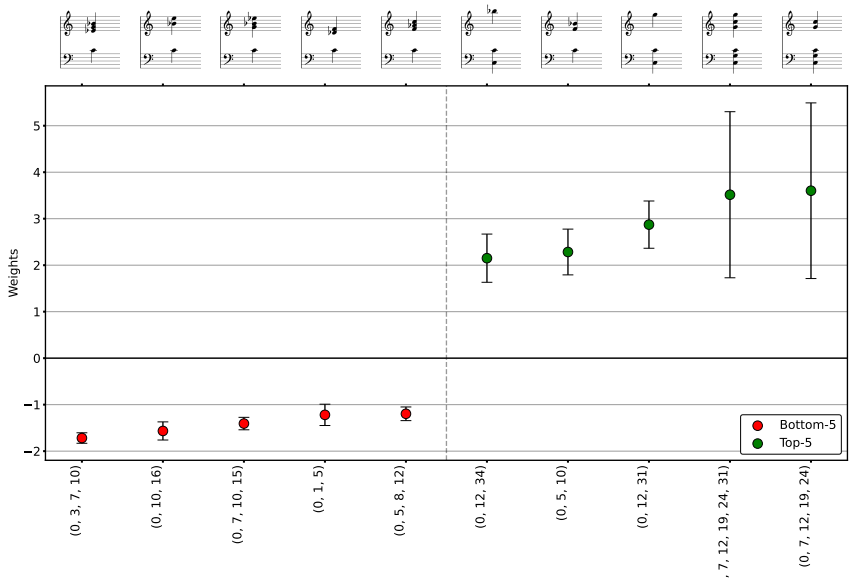
Supplementary Figure S.31: Predictive harmony features, Hank Jones.



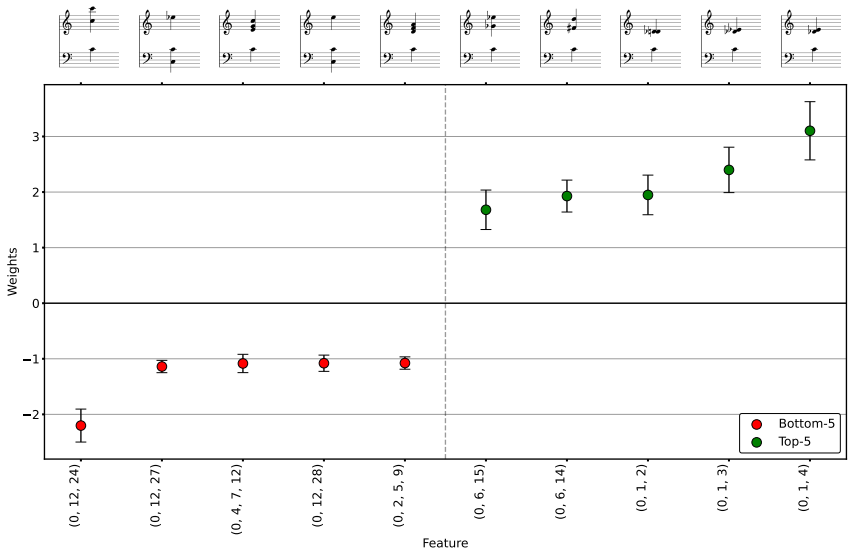
Supplementary Figure S.32: Predictive harmony features, John Hicks.



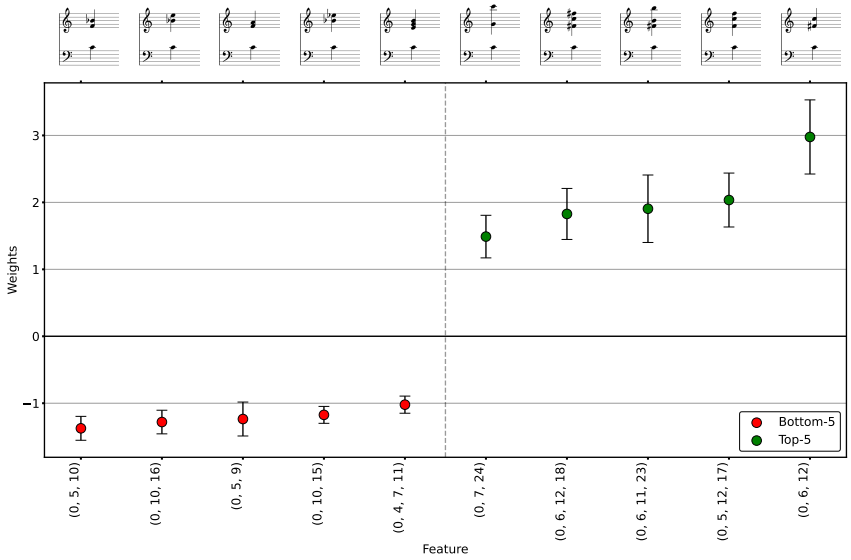
Supplementary Figure S.33: Predictive harmony features, Junior Mance.



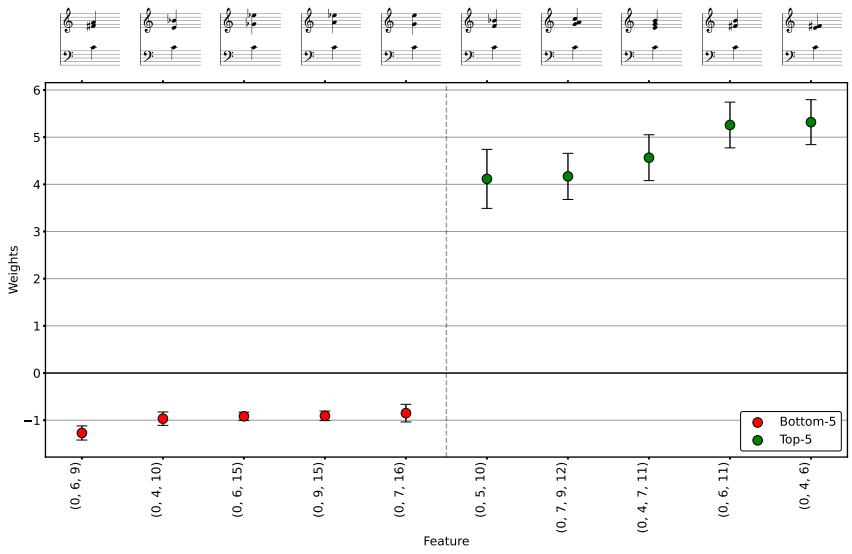
Supplementary Figure S.34: Predictive harmony features, Keith Jarrett.



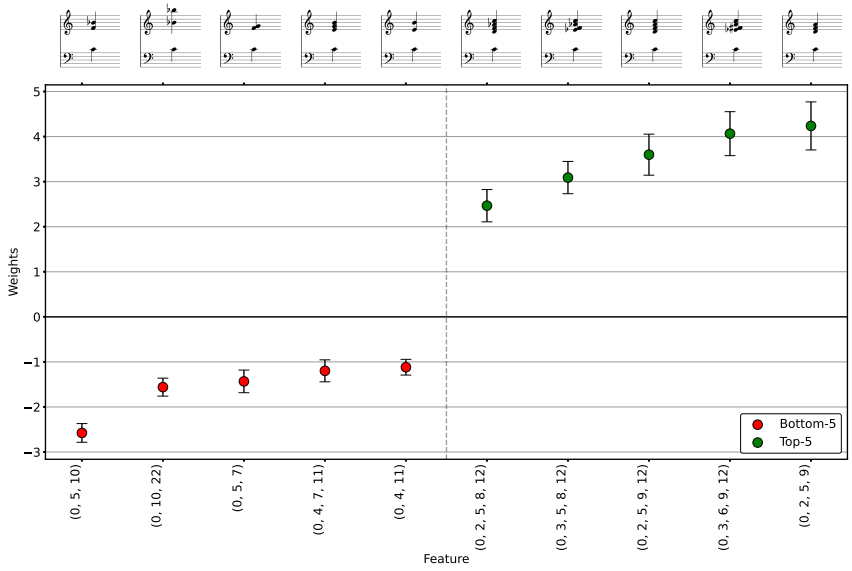
Supplementary Figure S.35: Predictive harmony features, Kenny Barron.



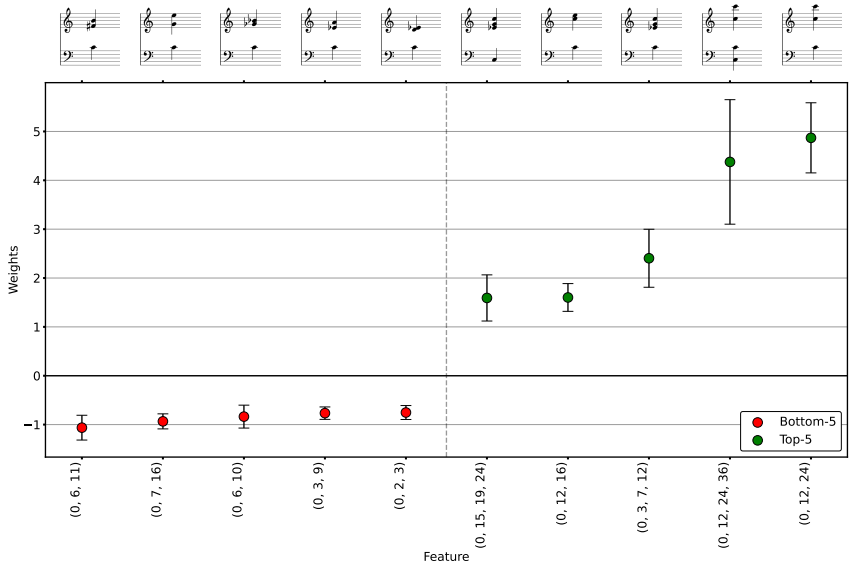
Supplementary Figure S.36: Predictive harmony features, Kenny Drew.



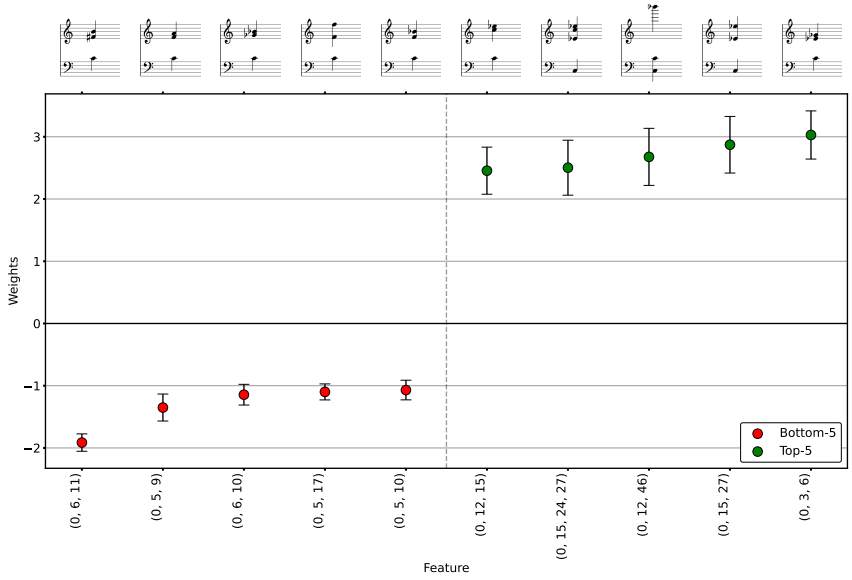
Supplementary Figure S.37: Predictive harmony features, McCoy Tyner.



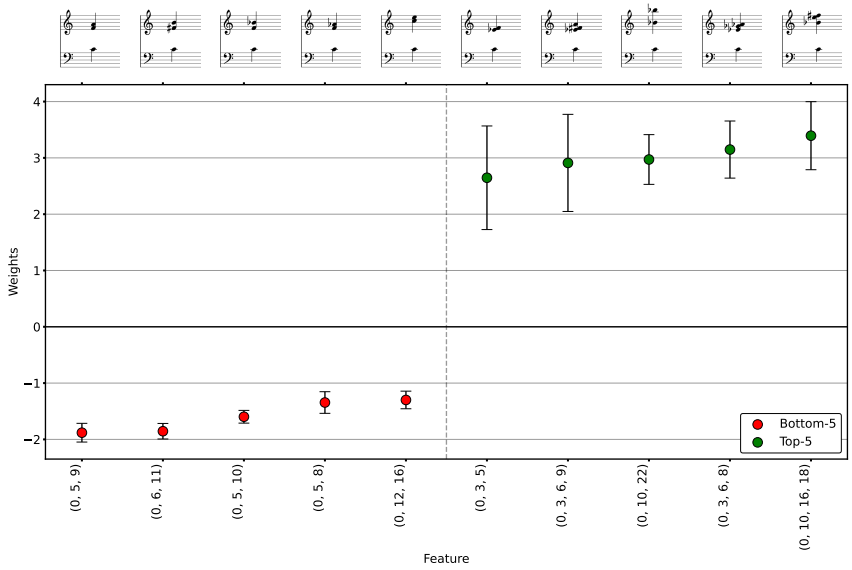
Supplementary Figure S.38: Predictive harmony features, Oscar Peterson.



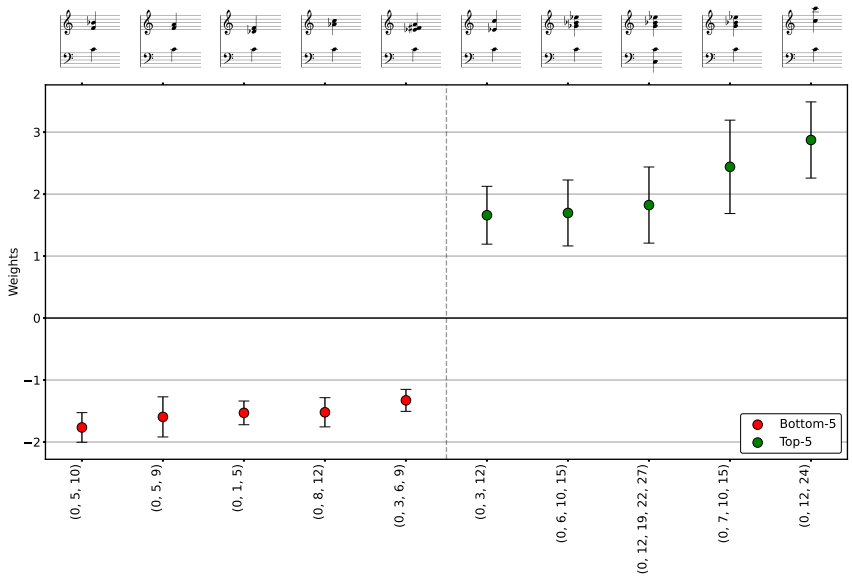
Supplementary Figure S.39: Predictive harmony features, Stanley Cowell.



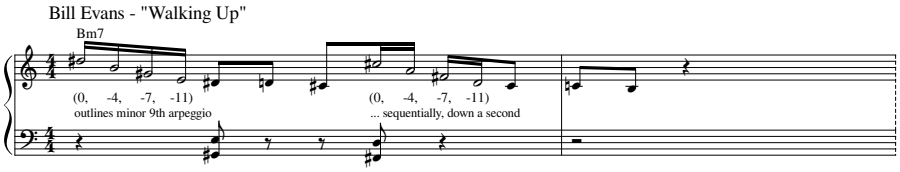
Supplementary Figure S.40: Predictive harmony features, Teddy Wilson.



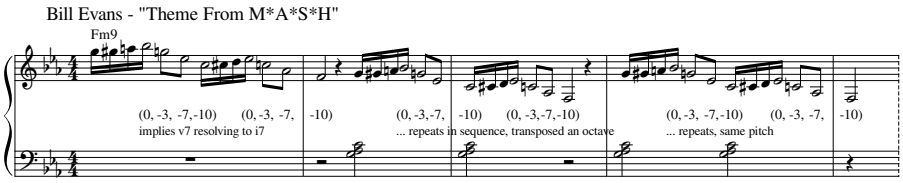
Supplementary Figure S.41: Predictive harmony features, Thelonious Monk.



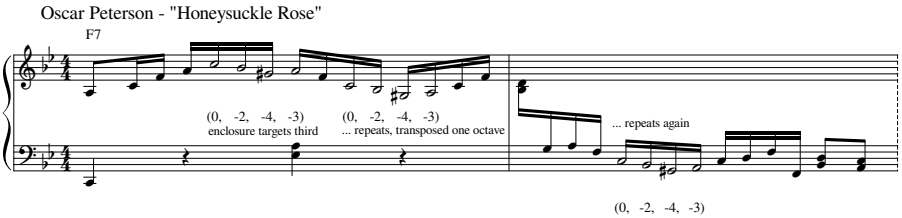
Supplementary Figure S.42: Predictive harmony features, Tommy Flanagan.



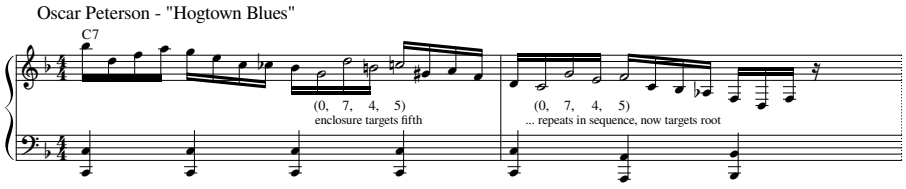
Supplementary Figure S.43: Bill Evans, usage of the $(0, -4, -7, -11)$ melodic pattern. The pattern is used to outline two descending minor ninth arpeggios, beginning on the ninth and descending to the root. The pattern appears twice in sequence to outline C# and B minor ninth arpeggios, respectively. Transcribed by the first author from a recording of "Walking Up" (1964).



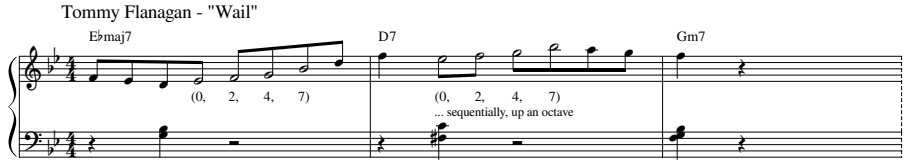
Supplementary Figure S.44: Bill Evans, usage of the $(0, -3, -7, -10)$ melodic pattern. The pattern is used in a sequence, outlining a minor seventh arpeggio built first on the fifth, then on the root (i.e., a perfect cadence). This process repeats three times (i.e., six appearances of the pattern). Each time the pattern appears, it is prefaced by brief chromatic run. Transcribed from a recording of "Theme from M*A*S*H" (1980).



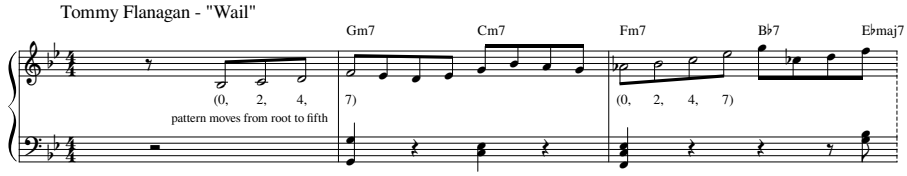
Supplementary Figure S.45: Oscar Peterson, usage of the $(0, -2, -4, -3)$ melodic pattern. The pattern is used to as an "enclosure" around the major third of the scale, approaching this chromatically from above and then below. The pattern appears multiple times in sequence, an octave lower each time. Transcribed from a recording of "Honeysuckle Rose" (1971).



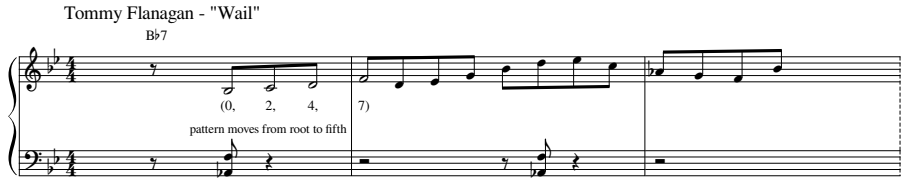
Supplementary Figure S.46: Oscar Peterson, usage of the $(0, 7, 4, 5)$ melodic pattern. The pattern is used to as an "enclosure": first, around the fifth of the scale; second, around the root. Transcribed from a recording of "Hogtown Blues" (1979).



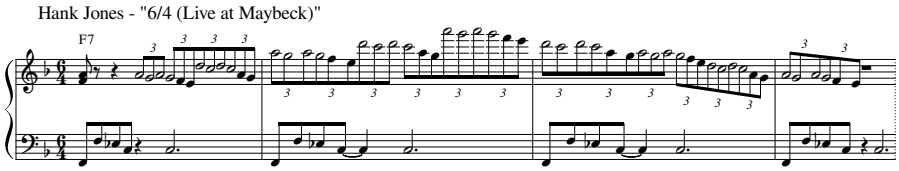
Supplementary Figure S.47: Tommy Flanagan, usage of the $(0, 2, 4, 7)$ melodic pattern. The pattern is used twice in sequence (an octave higher the second time), built from the fourth degree of the scale. Transcribed from a recording of "Wail" (1974).



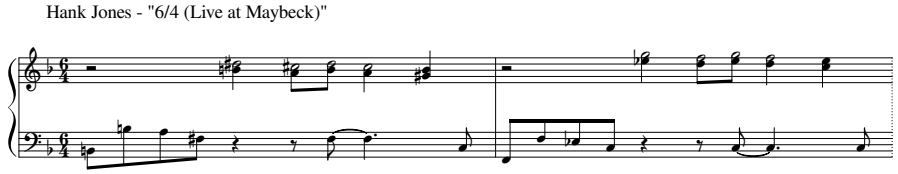
Supplementary Figure S.48: Tommy Flanagan, usage of the (0, 2, 4, 7) melodic pattern. Transcribed from a recording of "Wail" (1974).



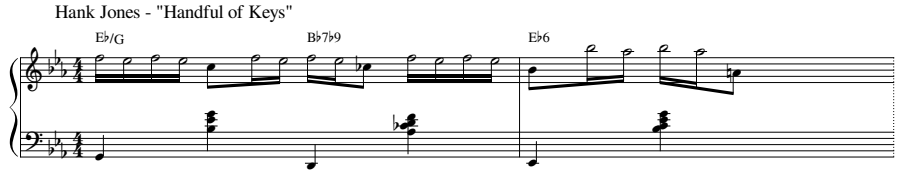
Supplementary Figure S.49: Tommy Flanagan, usage of the $(0, 2, 4, 7)$ melodic pattern. Transcribed from a recording of "Wail" (1974).



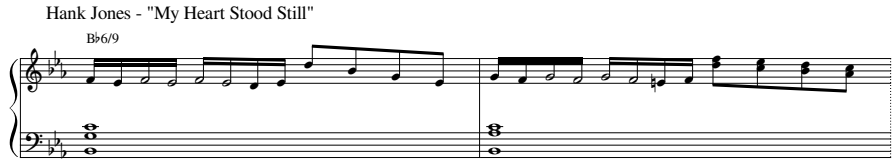
Supplementary Figure S.50: Hank Jones, usage of the $(0, -2, 0, -2)$ melodic pattern. The pattern forms part of a repeating sequence of six notes starting on either the third or sixth of the scale. Transcribed from a recording of "6/4" (1991).



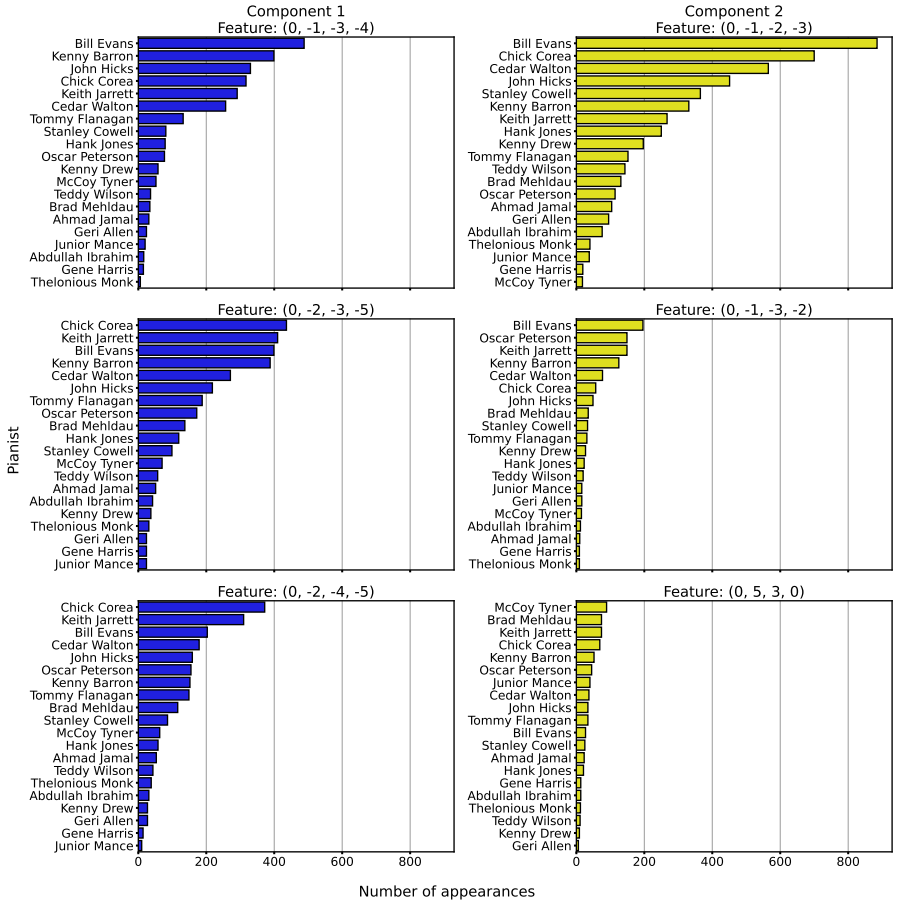
Supplementary Figure S.51: Hank Jones, usage of the $(0, -2, 0, -2)$ melodic pattern. The pattern forms part of a five-note figure, transposed up a diminished fourth to fit with the underlying harmony. Transcribed from a recording of "6/4" (1991).



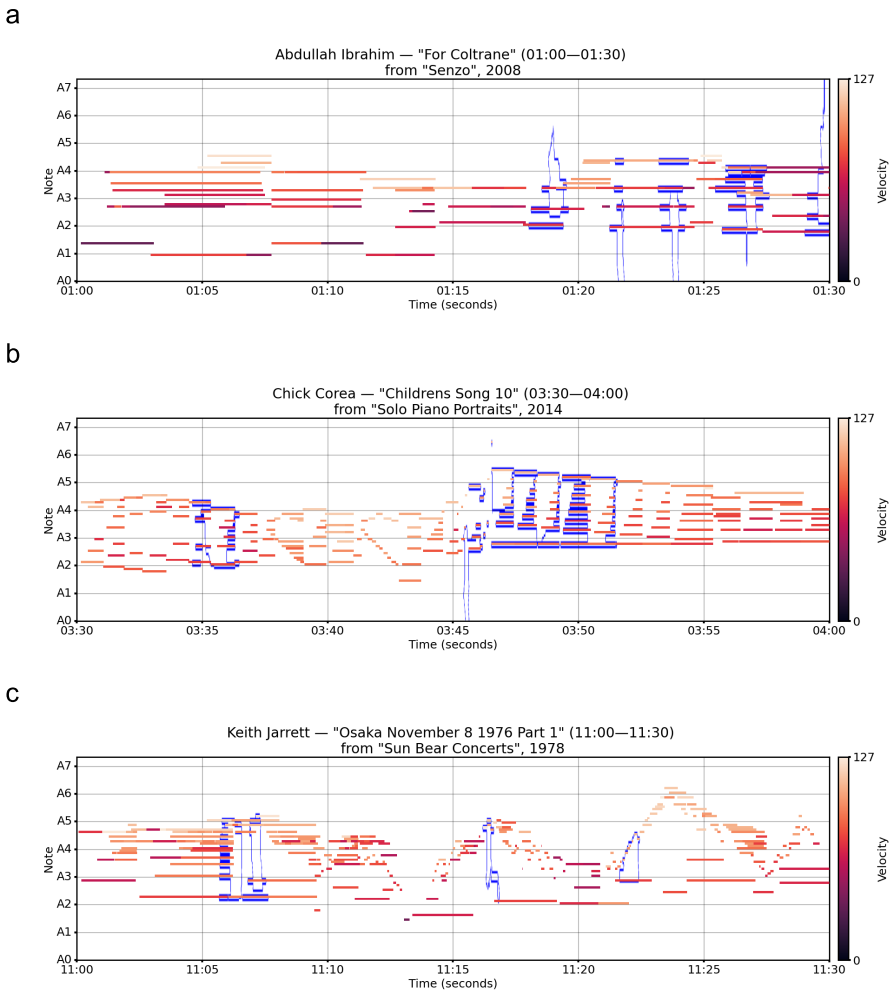
Supplementary Figure S.52: Hank Jones, usage of the $(0, -2, 0, -2)$ melodic pattern. The pattern is used multiple times in sequence with the lower note stepping chromatically down from the sixth of the scale to the fifth. Transcribed from a recording of "Handful of Keys" (1992).



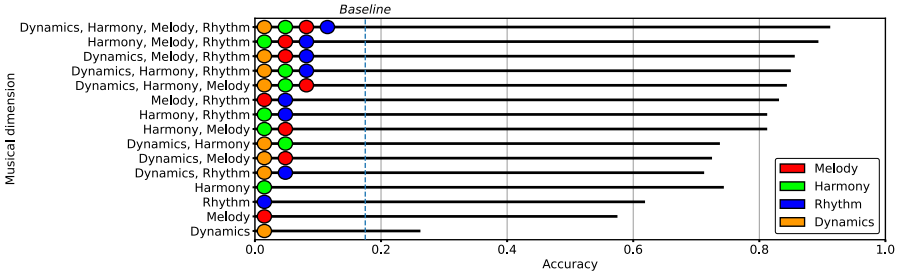
Supplementary Figure S.53: Hank Jones, usage of the $(0, -2, 0, -2)$ melodic pattern. Transcribed from a recording of "My Heart Stood Still" (1976).



Supplementary Figure S.54: PCA components and melodic pattern usage. The left column of panels in this figure shows the five melodic patterns that load the most onto Component 1 of the PCA analysis (Figure 3, left panel, x-axis). The right column shows the patterns that load most onto Component 2 (Figure 3, left panel, y-axis). The bars of each panel in this figure show the number of times each pattern appears in the recordings by every performer in our dataset.



Supplementary Figure S.55: Masked piano rolls generated with LIME. Each panel shows a single 30-second clip from a transcription of a performance by (a) Abdullah Ibrahim, (b) Chick Corea, and (c) Keith Jarrett, taken from the held-out test split. Highlighted in blue are the top five areas of the piano roll that positively contribute to predicting the target label according to LIME.



Supplementary Figure S.56: Multi-input model evaluation. The "lollipop" plot shows the accuracy obtained from predicting using all combinations of sub-networks, with the colored "heads" of each lollipop indicating the particular networks used to make the predictions. The dotted "baseline" score is the accuracy obtained from a model that simply predicts the majority class for every recording.

S.3 Supplementary Tables

Supplementary Table S.1: Optimised parameters (RF). These parameters resulted from the randomised search used to select parameters for the random forest classifier. Note that, when values for a parameter are sampled from either a uniform distribution $\mathcal{U}$ or logarithmic distribution $\mathcal{L}$ (with base 10), these are given in the range (start, stop).

Parameter	**Values**	**Description**	**Result**
`n_estimators`	$\sim \mathcal{U}(10, 401)$	Number of trees	265
`max_depth`	$\sim \mathcal{U}(1, 41)$	Maximum depth of each tree	26
`max_features`	$\sim \mathcal{U}(0.0001, 1)$	Proportion of features for best split	0.148
`bootstrap`	{True, False}	Whether or not to use bootstrap samples	False
`min_samples_leaf`	$\sim \mathcal{U}(1, 11)$	Minimum samples per leaf	3
`min_samples_split`	$\sim \mathcal{U}(2, 11)$	Minimum samples to split a node	7

Supplementary Table S.2: Optimised parameters (SVM).

Parameter	**Values**	**Description**	**Result**
`C`	$\sim \mathcal{L}(0.001, 1000)$	Regularization parameter: smaller values imply stronger regularization	1.359
`class_weight`	{None, balanced}	If balanced, adjusts weights inversely proportional to class frequencies	None

Supplementary Table S.3: Optimised parameters (LR).

Parameter	**Values**	**Description**	**Result**
`C`	$\sim \mathcal{L}(0.001, 1000)$	Identical to SVM	37.017
`class_weight`	{None, balanced}	Identical to SVM	balanced
`penalty`	{None, L_2}	Penalty term to use	L_2

References

1. Krumhansl, C. L. *Cognitive foundations of musical pitch* (Oxford University Press, 2001).
2. Riley, X. & Dixon, S. *Reconstructing the Charlie Parker Omnibook using an audio-to-score automatic transcription pipeline* in *Proceedings of the 21st Sound and Music Computing Conference* (Porto, Portugal, 2024), 546–553.
3. Cheston, H., Schlichting, J. L., Cross, I. & Harrison, P. M. C. Jazz trio database: Automated annotation of jazz piano trio recordings processed using audio source separation. *Transactions of the International Society for Music Information Retrieval* **7,** 144–158. `https://doi.org/10.5334/tismir.186` (2024).
4. Levine, M. *The jazz theory book* (Sher Music, Sebastopol, Ukraine, 2011).
5. Bittner, R. M., Bosch, J. J., Rubinstein, D., Meseguer-Brocal, G. & Ewert, S. *A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation* in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* ICASSP 2022 - 20222 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Singapore, 2022).
6. Kong, Q., Li, B., Song, X., Wan, Y. & Wang, Y. High-resolution piano transcription with pedals by regressing onset and offset times. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **29,** 3707–3717. ISSN: 2329-9290, 2329-9304. `https://ieeexplore.ieee.org/document/9585550/` (2021).
7. Schuller, G. in *Jazz panorama* (ed Williams, M. T.) 216–239 (The Jazz Book Club, London, 1965).
8. Frieler, K., Höger, F., Pfleiderer, M. & Dixon, S. *Two web applications for exploring melodic patterns in jazz solos* in *Proceedings of the 19th International Society for Music Information Retrieval Conference* (Paris, France, 2018).
9. Sidran, B. *Talking jazz: An illustrated oral history* (Pomegranate Artbooks, Petaluma, CA, 1992).
10. Levine, M. *The jazz piano book* ISBN: 978-1-4571-0144-1 (Sher Music, Sebastopol, Ukraine, 2011).
11. Shapiro, H. & Glebbeek, C. *Jimi Hendrix: Electric Gypsy* (St. Martin's Press, 1995).
12. Rodríguez-Algarra, F., Sturm, B. L. & Dixon, S. Characterising confounding effects in music classification experiments through interventions. *Transactions of the International Society for Music Information Retrieval* **2,** 52. ISSN: 2514-3298. `https://transactions.ismir.net/article/10.5334/tismir.24/` (2019).

13. Zhang, H., Karystinaios, E., Dixon, S., Widmer, G. & Cancino-Chacón, C. E. *Symbolic music representations for classification tasks: A systematic evaluation* in *Proceedings of the 24th International Society for Music Information Retrieval Conference* International Society for Music Information Retrieval Conference (ISMIR) (Milan, Italy, 2023).

14. Edwards, D., Dixon, S. & Benetos, E. PiJAMA: Piano jazz with automatic MIDI annotations. *Transactions of the International Society for Music Information Retrieval* **6,** 89–102. ISSN: 2514-3298. http://transactions.ismir.net/articles/10.5334/tismir.162/ (2023).

15. Reimers, N. & Gurevych, I. *Sentence-BERT: Sentence embeddings using Siamese BERT-Networks* in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, 2019). https://arxiv.org/abs/1908.10084.

16. Gioia, T. *The history of jazz* 2nd ed. 445 pp. (Oxford University Press, New York, 2011).

17. Bunks, C., Weyde, T., Dixon, S. & Giorgi, B. D. *Modeling harmonic similarity for jazz using co-occurrence vectors and the membrane area* in *Proceedings of the 24th International Society for Music Information Retrieval Conference* (Milan, Italy, 2023), 757–764.

18. Norgaard, M., Bales, K. & Hansen, N. C. Linked auditory and motor patterns in the improvisation vocabulary of an artist-level jazz pianist. *Cognition* **230.** https://www.sciencedirect.com/science/article/pii/S0010027722002967 (2023).

19. Mahmud Rafee, S. R., Fazekas, G. & Wiggins, G. *HIPI: A hierarchical performer identification model based on symbolic representation of music* in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (IEEE, Rhodes Island, Greece, 2023), 1–5. ISBN: 978-1-72816-327-7. https://ieeexplore.ieee.org/document/10094844/.

20. Kong, Q., Choi, K. & Wang, Y. *Large-scale MIDI-based composer classification* 2020. arXiv: 2010.14805[cs,eess]. http://arxiv.org/abs/2010.14805.

21. Kim, S., Lee, H., Park, S., Lee, J. & Choi, K. *Deep composer classification using symbolic representation* in *Extended Abstracts for the Late-Breaking Demo Session of the 21st International Society for Music Information Retrieval Conference* (Montreal, Canada, 2020).

22. Foscarin, F., Hoedt, K., Praher, V., Flexer, A. & Widmer, G. *Concept-based techniques for "musicologist-friendly" explanations in a deep music classifier* in *Proceedings of the 23rd International Society for Music Information Retrieval Conference* (Bengaluru, India, 2022).

23. Paszke, A. *et al. PyTorch: An imperative style, high-performance deep learning library* in *Advances in Neural Information Processing Systems 32* (Vancouver, Canada, 2019).

24. Liu, L., Kong, Q., Morfi, V. & Benetos, E. *Performance MIDI-to-score conversion by neural beat tracking* in *Proceedings of the 23rd International Society for Music Information Retrieval Conference* (Bengaluru, India, 2022).