Deep Neural Models for Jazz Improvisations

Shunit Haviv Hakimi
Deep Neural Models for Jazz Improvisations

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Shunit Haviv Hakimi

Submitted to the Senate
of the Technion — Israel Institute of Technology
Kislev 5781 Haifa November 2020
This research was carried out under the supervision of Prof. Ran El-Yaniv, in the Faculty of Computer Science.

Some results in this thesis have been published as an article by the author and research collaborators in a conference during the course of the author’s research period, the most up-to-date version of which being:


The generous financial help of the Technion is gratefully acknowledged.
# Contents

## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

## Abstract

## 1 Introduction

## 2 BebopNet: Deep Neural Models for Personalized Jazz Improvisations

2.1 Introduction ................................................. 7
2.2 Related Work ................................................. 8
2.3 Problem Statement ............................................ 9
2.4 Methods ....................................................... 10
  2.4.1 BebopNet: Jazz Model Learning ............................... 10
  2.4.2 User Preference Elicitation ................................ 12
  2.4.3 User Preference Metric Learning ............................. 12
  2.4.4 Optimized Music Generation ................................ 13
2.5 Experiments ................................................... 14
  2.5.1 Harmonic Coherence ......................................... 14
  2.5.2 Analyzing Personalized Models ............................... 14
2.6 Plagiarism Analysis ........................................... 16

## 3 Musical Features

3.1 Introduction ................................................... 17
3.2 Musical Feature List ........................................... 17
  3.2.1 Number of notes ............................................ 18
  3.2.2 Duration entropy ........................................... 18
  3.2.3 Pentatonic scale ........................................... 18
  3.2.4 Blues scale ................................................ 18
  3.2.5 Syncopation ............................................... 19
  3.2.6 Chord note on a downbeat .................................. 19
  3.2.7 Chromatic before a chord note on a down-beat ............... 19
  3.2.8 Chord match ............................................... 19
  3.2.9 Scale match ................................................ 20
  3.2.10 Altered scale match ....................................... 20
# List of Figures

1.1 A short excerpt generated by BebopNet. ........................................ 4

2.1 An example of a measure in music notation and its vector representation. Integers are converted to one-hot representations. .......................... 11

2.2 The BebopNet architecture for the next note prediction. Each note is represented by concatenating the embeddings of the pitch (red bar), the duration (purple bar) and the four pitches comprising the current chord (green bars). The output of the LSTM is passed to two heads (orange bars), one the size of the pitch embedding (top) and the other the size of the duration embedding (bottom). ........................................ 12

2.3 2.3i Predictions of the preference model on sequences from a validation set. Green: sequences labeled with a positive score ($y_r > 0$); yellow: neutral ($y_r = 0$); red: negative ($y_r < 0$). The blue vertical lines indicate thresholds $\beta_1, \beta_2$ used for selective prediction. 2.3ii Risk-coverage plot for the predictions of the preference model. $\beta_1, \beta_2$ (green lines) are defined to be the thresholds that yield a minimum error on the contour of 25% coverage. .................................................. 16

3.1 Feature distribution over the dataset (histogram (bars), estimated distribution (lines)) .......................................................... 22

4.1 Active Learning vs. Passive Learning convergence checks: 1-hot feature target preferred. The graphs are an average of 10 runs, and the confidence interval is for 95%. ........................................ 28

4.2 Active Learning vs. Passive Learning convergence checks: 5-hot feature target preferred. The graphs are an average of 10 runs, and the confidence interval is for 95%. ........................................ 29

4.3 Active Learning vs. Passive Learning convergence checks: Manually labeled dataset ............................................................. 29

5.1 The loss of the pre-trained LSTM model (BebopNet for RL) .............. 36

5.2 The accuracy of the pitch of the estimated next note (by BebopNet for RL) ............................................................................ 36
5.3 The accuracy of the duration of the estimated next note (by BebopNet for RL) ............................................................ 36

5.4 The complete model architecture we used. (model) contains the pre-trained LSTM, (rnn) is the additional LSTM and (actor_base), (actor_pitch), (actor_dur), (critic) and (critic_linear) are feed-forward layers dedicated to the A2C predictions. ................................................ 37

5.5 Example of the convergence of the reward during the training process. . 38

7.1 Digital CRDI controlled by a user to provide continuous preference feedback. ............................................................... 44

7.2 User 1: 7.2i Predictions of the preference model on sequences from a validation set. Green: sequences labeled with a positive score \( y_\tau > 0 \); yellow: neutral \( y_\tau = 0 \); red: negative \( y_\tau < 0 \). Blue vertical line indicates thresholds \( \beta_1, \beta_2 \) used for selective prediction. 7.2ii Risk-coverage plot for the predictions of the preference model. \( \beta_1, \beta_2 \) (green lines) are defined to be the thresholds that yield minimum error on the contour of 25% coverage. .............................................................. 48

7.3 User 2: 7.3i Predictions of the preference model on sequences from a validation set. Green: sequences labeled with a positive score \( y_\tau > 0 \); yellow: neutral \( y_\tau = 0 \); red: negative \( y_\tau < 0 \). Blue vertical line indicates thresholds \( \beta_1, \beta_2 \) used for selective prediction. 7.3ii Risk-coverage plot for the predictions of the preference model. \( \beta_1, \beta_2 \) (green lines) are defined to be the thresholds that yield minimum error on the contour of 25% coverage. .............................................................. 48

7.4 User 3: 7.4i Predictions of the preference model on sequences from a validation set. Green: sequences labeled with a positive score \( y_\tau > 0 \); yellow: neutral \( y_\tau = 0 \); red: negative \( y_\tau < 0 \). Blue vertical line indicates thresholds \( \beta_1, \beta_2 \) used for selective prediction. 7.4ii Risk-coverage plot for the predictions of the preference model. \( \beta_1, \beta_2 \) (green lines) are defined to be the thresholds that yield minimum error on the contour of 25% coverage. .............................................................. 49

7.5 User 4: 7.5i Predictions of the preference model on sequences from a validation set. Green: sequences labeled with a positive score \( y_\tau > 0 \); yellow: neutral \( y_\tau = 0 \); red: negative \( y_\tau < 0 \). Blue vertical line indicates thresholds \( \beta_1, \beta_2 \) used for selective prediction. 7.5ii Risk-coverage plot for the predictions of the preference model. \( \beta_1, \beta_2 \) (green lines) are defined to be the thresholds that yield minimum error on the contour of 25% coverage. .............................................................. 49
7.6 User Score vs. Beam Width: As we increase the width of the beam, we get a higher score for generated solos using the user preference model. x-axis - b\_k combinations of beam width b and parameter k used. Shaded area represents the 95\% percentile of the confidence interval. Notice the initial point of beam width 1\_1 representing the score for improvisations generated by BebopNet without personalization.

7.7 Percent of common phrases of length n-gram length. Our jazz model is in black (BebopNet).
Abstract

A major bottleneck in the evaluation of music generation is that music appreciation is a highly subjective matter. When considering an average appreciation as an evaluation metric, user studies can be helpful. The challenge of generating personalized content, however, has been examined only rarely in the literature. In this work, we address generation of personalized music and propose a novel pipeline for music generation that learns and optimizes user-specific musical taste. We focus on the task of symbol-based, monophonic, harmony-constrained jazz improvisations. Our personalization pipeline begins with BebopNet, a music language model trained on a corpus of jazz improvisations by Bebop giants. BebopNet is able to generate improvisations based on any given chord progression. We then assemble a personalized dataset, labeled by a specific user, and train a user-specific preference metric that reflects this user’s unique musical taste. Finally, we employ a personalized variant of beam-search with BebopNet to optimize the generated jazz improvisations for that user. We present an extensive empirical study in which we apply this pipeline to extract individual models as implicitly defined by several human listeners. Our approach enables an objective examination of subjective personalized models whose performance is quantifiable. The results indicate that it is possible to model and optimize personal jazz preferences and offer a foundation for future research in personalized generation of art. We further extend this generation method and present feature-guided improvisation generation that allows users to define a combination of musical features for BebopNet to optimize. For this purpose, we define numerous measurable musical features inspired by jazz music theory. Additionally, we inspect the possible benefits of using active learning to learn the user-specific preference metric, and the possibility of replacing our optimization step in the personalization pipeline with reinforcement learning. We discuss our experiments, results, and conclusions. We also briefly discuss opportunities, challenges, and questions that arise from our work, including issues related to creativity.
Chapter 1

Introduction

Since the dawn of computers, researchers and artists have been interested in utilizing them for producing different forms of art, and notably for composing music [HJI57]. The explosive growth of deep learning models over the past several years has expanded the possibilities for musical generation, leading to a line of work that pushed forward the state-of-the-art [JGTE17, HVU+18, HSR+18, CZD+19, CGRS19]. Another recent trend is the development and offerings of consumer services such as Spotify, Deezer and Pandora, aiming to provide personalized streams of existing music content. Perhaps the crowning achievement of such personalized services would be for the content itself to be generated explicitly to match each individual user’s taste. In this work we focus on the task of generating user personalized, monophonic, symbolic jazz improvisations. To the best of our knowledge, this is the first work that aims at generating personalized jazz solos using deep learning techniques.

The common approach for generating music with neural networks is generally the same as for language modeling. Given a context of existing symbols (e.g., characters, words, music notes), the network is trained to predict the next symbol. Thus, once the network learns the distribution of sequences from the training set, it can generate novel sequences by sampling from the network output and feeding the result back into itself. The products of such models are sometimes evaluated through user studies (crowd-sourcing). Such studies assess the quality of generated music by asking users their opinion, and computing the mean opinion score (MOS). While these methods may measure the overall quality of the generated music, they tend to average-out evaluators’ personal preferences. Another, more quantitative and rigid approach for evaluation of generated music is to compute a metric based on musical theory principles. While such metrics can, in principle, be defined for evaluating classical music, they are less suitable for jazz improvisation, which does not adhere to such strict rules. That being said, there are musical metrics that can be useful for analyzing and describing generated music. For example, musical properties may help us explain and distinguish the differences between jazz improvisations. Similarly to image descriptors [KS04, MS05], such musical properties enable a featured representation of jazz improvisations in a feature space of
a lower dimension.

As many jazz experts would recommend, the key to attaining great improvisation skills is by studying and emulating great musicians. Following this advice, we train BebopNet, a harmony-conditioned jazz model that composes entire solos. We use a training dataset of hundreds of professionally transcribed jazz improvisations performed by saxophone giants such as Charlie Parker, Phil Woods and Cannonball Adderley (see details in Section 2.4.1). After the training process, BebopNet is capable of generating high fidelity improvisation phrases (this is a subjective impression of the authors). Figure 1.1 presents a short excerpt generated by BebopNet.

One of the main difficulties in evaluating music is that different people have different musical tastes. Therefore, our goal in Chapter 2 is to go beyond straightforward generation by BebopNet and optimize the generation toward personalized preferences. To this end, we propose a pipeline for personalized jazz improvisations generation consisting of the following elements: (a) BebopNet: jazz model learning; (b) user preference elicitation; (c) user preference metric learning; and (d) optimized music generation via planning.

After the training of BebopNet, we train a personal preference metric that predicts a certain user’s musical preference. To this end, we perform a user preference elicitation process to construct a personal dataset consisting of jazz improvisations and the user preference for these improvisations. The user tags their preference using a meter inspired by the CRDI (continuous response digital interface) device [Rob88] while listening to the improvisations. After collecting sufficient data (see Section 2.4.2), we train a personal user preference metric. Having a trained user preference metric, in the final stage of our pipeline, we employ beam-search [Nor92] over BebopNet to optimize the generated improvisations according to the user’s preference metric. The beam-search uses the criterion of the score of the user preference metric, and thus allows BebopNet to generate personalized improvisations that match the user’s musical taste.

While the results described in Chapter 2 indicate that our proposed pipeline enables personalized improvisation generation, it raises questions regarding the musical features learned by the user preference metric. To what extent does our metric express the specifics of one’s musical taste? Can we extract precise musical properties from this metric? These issues drove us towards researching a more explainable model and developing a method for controlling the specific musical features of a generated improvisation. To this aim, in Chapter 3, we present a list of musical features. These features allow us to analyze jazz improvisations as well as to guide our improvisation...
generation towards an explicit setting of musical features. To create feature-guided improvisations, we employ the final step of our personalization pipeline. Here, instead of the user preference metric, we use beam-search with a defined feature setting as an optimization target.

Having defined these musical features, we can now use the resulting feature space to represent jazz improvisations. Here, we train the user preference metric on the musical features space representation of the jazz improvisations. In this setting, the user’s preference metric makes use of the musical feature space. This increases our ability to musically analyze the user’s preference metric as well as to compare the musical features expressed by different users’ personal preference metrics. In Chapter 4, we utilize our feature space to optimize the process of personal preference metric learning with active learning [Set09]. Active learning is designed to interactively query an information source for labels, choosing the most informative queries to benefit the learning process. We present a study considering active learning vs. passive learning for preference metric learning. We examine both an automatically labeled dataset of a specific musical feature as a target preference, and a human labeled preference dataset. We present our results and discuss the implications and difficulties of utilizing active learning on a human labeled preference dataset.

Finally, in Chapter 5, we consider an alternative approach for optimized improvisation generation with reinforcement learning. Instead of using beam-search over a base model (BebopNet), we train a new optimized generative model using reinforcement learning. For this task, we create a new reinforcement learning environment for jazz improvisation generation. We start our experiments by defining a feature-based reward. Thus, our generative model is trained to optimize a certain setting of musical features. Additionally, we train a personalized generative model with reinforcement learning that optimizes a certain user’s musical taste. To this end, we set the learned user preference metric as our reward function. We describe our results and discuss the difficulties in reinforcement learning for human preferences.
Chapter 2

BebopNet: Deep Neural Models for Personalized Jazz Improvisations

2.1 Introduction

To generate personalized jazz improvisations, we propose a pipeline consisting of the following elements: (a) BebopNet: jazz model learning; (b) user preference elicitation; (c) user preference metric learning; and (d) optimized music generation via planning.

We start our pipeline by training BebopNet, a harmony-conditioned jazz model that composes entire solos. We use a training dataset of hundreds of professionally transcribed jazz improvisations performed by saxophone giants such as Charlie Parker, Phil Woods and Cannonball Adderley (see details in Section 2.4.1). In this dataset, each solo is a monophonic note sequence given in symbolic form (MusicXML) accompanied by a synchronized harmony sequence. After training, BebopNet is capable of generating improvisation phrases to any given chord sequence.

Considering that different people have different musical tastes, our goal in this chapter is to go beyond straightforward generation by this model and optimize the generation toward personalized preferences. For this purpose, we determine a user’s preference by measuring the level of their satisfaction throughout the solos using a digital variant of continuous response interface (CRDI)[Rob88]. This is accomplished by playing, for the user, computer-generated solos (from the jazz model) and recording their good/bad feedback in real time throughout each solo. Once we have gathered sufficient data about the user’s preferences, consisting of two aligned sequences (for the solos and feedback), we train a user preference metric in the form of a recurrent regression model to predict this user’s preferences. A key feature of our technique is that the resulting model can be evaluated objectively using hold-out user preference sequences (along with their corresponding solos). A big hurdle in accomplishing this...
step is that the signal elicited from the user is inevitably extremely noisy. To reduce this noise, we apply selective prediction techniques [GEY17, GEY19] to distill cleaner predictions from the user’s preference model. Thus, we allow this model to abstain whenever it is not sufficiently confident. The fact that it is possible to extract a human continuous response preference signal on musical phrases and use it to train (and test) a model with non-trivial predictive capabilities is interesting in itself (and new, to the best of our knowledge).

Equipped with a personalized user preference metric (via the trained model), in the last stage we employ a variant of beam-search [Nor92], to generate optimized jazz solos from BebopNet. For each user, we apply the last three stages of this process where the preference elicitation stage takes several hours of tagging per user. We applied the proposed pipeline on four users, all of whom are amateur jazz musicians. We present numerical analysis of the results showing that a personalized metric can be trained and then used to optimize solo generation.

To summarize, our contributions include: (1) a useful monophonic neural model for general jazz improvisation within any desired harmonic context; (2) a viable methodology for eliciting and learning high resolution human preferences for music; (3) a personalized optimization process of jazz solo generation; and (4) an objective evaluation method for subjective content and plagiarism analysis for the generated improvisations.

2.2 Related Work

Many different techniques for algorithmic musical composition have been used over the years. For example, some are grammar-based [GTK09], rule-based [HJI57, Löt99], use Markov chains [Pac03, WB05, STLP17], evolutionary methods [LK94, PW98] or neural networks [Toi95, NW92, Fra01]. For a comprehensive summary of this broad area, we refer the reader to [FV13]. Here we confine the discussion to closely related works that mainly concern jazz improvisation using deep learning techniques over symbolic data. In this narrower context, most works follow a generation by prediction paradigm, whereby a model trained to predict the next symbol is used to greedily generate sequences. The first work on blues improvisation [ES02] straightforwardly applied long short-term memory (LSTM) networks on a small training set. While their results may seem limited at a distance of nearly two decades\(^1\), they were the first to demonstrate long-term structure captured by neural networks.

One approach to improving a naïve greedy generation from a jazz model is by using a mixture of experts. For example, Franklin et al. [Fra06] trained an ensemble of neural networks were trained, one specialized for each melody, and then selected from among them at generation time using reinforcement learning (RL) utilizing a hand-crafted reward function. Johnson et al. [JKW17] generated improvisations by training

\(^1\)Listen to their generated pieces at [www.iro.umontreal.ca/~eckdoug/blues/index.html](http://www.iro.umontreal.ca/~eckdoug/blues/index.html).
a network consisting of two experts, each focusing on a different note representation. The experts were combined using the technique of product of experts [Hin02]. Other remotely related non-jazz works have attempted to produce context-dependent melodies [JGTE17, CZD+19, YCY17, HVU+18, HSR+19, HPN17, MSC18, HN17].

A common method for collecting continuous measurements from human subjects listening to music is the continuous response digital interface (CRDI), first reported by [Rob88]. CRDI has been successful in measuring a variety of signals from humans such as emotional response [Sch01], tone quality and intonation [MG99], beauty in a vocal performance [Him11], preference for music of other cultures [Bri96] and appreciation of the aesthetics of jazz music [Cog04]. Using CRDI, listeners are required to rate different elements of the music by adjusting a dial (which looks similar to a volume control dial present on amplifiers).

2.3 Problem Statement

We now state the problem in mathematical terms. We denote an input \( x_t = (s_t, c_t) \) consisting of a note \( s_t \) and its context \( c_t \). Each note \( s_t \in S \), in turn, consists of a pitch and a duration at index \( t \) and \( S \) represents a predefined set of pitch-duration combinations (i.e., notes). The context \( c_t \in C \) represents the chord that is played with note \( s_t \), where \( C \) is the set of all possible chords. The context may contain additional information such as the offset of the note within a measure (see details in Section 2.4). Let \( D \) denote a training dataset consisting of \( M \) solos. Each solo is a sequence \( X_\tau = x_1 \cdots x_\tau \in (S \times C)^\tau \) of arbitrary length \( \tau \). In our work, these are the aforementioned jazz improvisations.

We define a context-dependent jazz model \( f_\theta \) (Eq. 2.1), as the estimator of the probability of a note \( s_t \) given the sequence of previous inputs \( X_{t-1} \) and the current context \( c_t \), where \( \theta \) are the parameters of the model. This is similar to a human jazz improviser who is informed of the chord over which his next note will be played.

\[
f_\theta(X_{t-1}, c_t) = Pr(s_t|X_{t-1}, c_t) \tag{2.1}
\]

For any solo \( X_\tau \), we also consider an associated sequence of annotation \( Y_\tau = y_1 \cdots y_\tau \in \mathcal{Y}^\tau \). An annotation \( y_t \in \mathcal{Y} \) represents the quality of the solo up to point \( t \) by some metric. In our case, \( y_t \) may be a measure of preference as indicated by a user or a score measuring harmonic compliance. Let \( \tilde{D} \) denote a training dataset consisting of \( N \) solos. Each solo \( X_\tau \) of arbitrary length \( \tau \) is labeled with a sequence \( Y_\tau \). Given \( \tilde{D} \), we define a metric \( g_\phi \) (Eq. 2.2) to predict \( y_\tau \) given a sequence of inputs \( X_\tau \). \( g_\phi \) is the user-preference model and \( \phi \) are the learned parameters.

\[
\hat{y}_\tau = g_\phi(X_\tau) \tag{2.2}
\]

\(^2\)Listen to the generated solos at www.cs.hmc.edu/~keller/jazz/improvisor/iccc2017/
We denote by $\psi$ a function that is used to sample notes from $f_\theta$ to generate solos. In our case, this will be our beam-search variant. The objective here is to train viable models, $f_\theta$ and $g_\phi$, and then to use $\psi$ to sample solos from $f_\theta$ while maximizing $g_\phi$.

2.4 Methods

In this section we describe the methods used and implementation details of our personalized generation pipeline.

2.4.1 BebopNet: Jazz Model Learning

In the first step of our pipeline, we use supervised learning to train BebopNet, a context-dependent jazz model $f_\theta$ from a given corpus of transcribed jazz solos.

Dataset and music representation

Our corpus $D$ consists of 284 professionally transcribed solos of (mostly) Bebop saxophone players of the early 20th century. These are Charlie Parker, Sonny Stitt, Cannonball Adderley, Dexter Gordon, Sonny Rollins, Stan Getz, Phil Woods and Gene Ammons. We consider only solos that are in 4/4 metre and include chords in their transcription. The solos are provided in musicXML format. As opposed to MIDI, this format allows the inclusion of chord symbols. We represent notes using a representation method inspired by sheet music (see Figure 2.1). After considering multiple representation options that try to exploit the continuous and circular properties of pitch and duration, we found that using a learned embedding layer for representation achieves better results.

**Pitch** The pitch is encoded as a one-hot vector of size 129. Indices 0—127 match the pitch range of the MIDI standard. Index 128 corresponds to the rest symbol.

**Duration** The duration of each note is encoded using a one-hot vector consisting of all the existing durations in the dataset. Durations smaller than 1/24 are removed.

**Offset** The offset of the note lies within the measure and is quantized to 48 “ticks” per (four-beat) measure. This corresponds to a duration of 1/12 of a beat. This is similar to the learned positional-encoding used in translation.

**Chord** The chord is represented by a four-hot vector of size 12, representing the 12 possible pitch classes to appear in a chord. As common in jazz music, unless otherwise noted, we assume that chords are played using their 7th form. Thus, the chord pitches are usually the 1st, 3rd, 5th, and 7th degrees of the root of the chord. This chord rep-

---

3 The solos were purchased from SaxSolos.com [sax]: we are thus unable to publish them. Nevertheless, in Section 7.5 of the appendix we provide a complete list of solos used for training, which are available from the above vendor.

4 The notes appearing in the corpus all belong to a much smaller range; however, the MIDI range standard was maintained for simplicity.
Figure 2.1: An example of a measure in music notation and its vector representation. Integers are converted to one-hot representations.

This representation allows the flexibility of representing rare chords such as sixth, diminished and augmented chords.

**Network Architecture**

BebopNet, as many language models, can be implemented using different architectures such as recurrent neural networks (RNNs), convolutional networks (CNNs) [vdODZ+16, YCY17, CZD+19] or attention-based models [ACC+18]. BebopNet contains a three-layer LSTM network [HS97]. Recent promising results with attention based models enabled us to improve BebopNet by replacing the LSTM with Transformer-XL [DYY+19]. This architectural change was performed after constructing and experimenting with the presented pipeline, and showed improvements in the structure and motifs captured by BebopNet. The architecture of the network used to estimate $f_\theta$ is illustrated in Figure 2.2. The network’s input $x_t$ includes the note $s_t$ (pitch and duration) and context $c_t$ (offset and chord). The pitch, duration and offset are each represented by learned embedding layers. The chord is encoded by using the embedding of the pitches comprising it. While notes at different octaves have different embeddings, the chord pitch embeddings are always taken from the octave in which most notes in the dataset reside. This embedded vector is passed to the LSTM network. The LSTM output is then passed to two heads. Each head consists of two fully-connected layers with a sigmoid activation in-between. The output of the first layer is the same size as the embedding of the pitch (or duration), and the second output size is the number of possible pitches (or durations). Following [IKS17, PW17], we tie the weights of the final fully-connected layers to those of the embedding. Finally, the outputs of the two heads pass through a softmax layer and are trained to minimize the negative log-likelihood of the corpus. To enrich our dataset while encouraging harmonic context dependence, we augment our dataset by transposing to all 12 keys.
Figure 2.2: The BebopNet architecture for the next note prediction. Each note is represented by concatenating the embeddings of the pitch (red bar), the duration (purple bar) and the four pitches comprising the current chord (green bars). The output of the LSTM is passed to two heads (orange bars), one the size of the pitch embedding (top) and the other the size of the duration embedding (bottom).

2.4.2 User Preference Elicitation

Using BebopNet, we created a dataset to be labeled by users, consisting of 124 improvisations. These solos were divided into three groups of roughly the same size: solos from the original corpus, solos generated by BebopNet over jazz standards present in the training set, and generated solos over jazz standards not present in the training set. The solos by BebopNet were generated by sampling the softmax distributions of the pitch and duration outputs note by note. The length of each solo is two choruses, or twice the length of the melody. For each standard, we generated a backing track in MP3 format that includes a rhythm section and a harmonic instrument to play along the improvisation using Band-in-a-Box [PG]. This dataset amounts to approximately five hours of played music.

We created a system inspired by CRDI that is entirely digital, replacing the analog dial with strokes of a keyboard moving a digital dial. A figure of our dial is presented in the appendix (Figure 7.1). While the original CRDI had a range of 255 values, our initial experiments found that quantizing the values to five levels was easier for users. We recorded the location of the dial at every time step and aligned it to the note being played at the same moment.

2.4.3 User Preference Metric Learning

In the user preference metric learning stage we again use supervised learning to train a metric function $g_\phi$. This function should predict user preference scores for any solo, given its harmonic context. During training, for each sequence $X_\tau$ we estimate $y_\tau$, corresponding to the label the user provided for the last note in the sequence. We choose the last label of the sequence, rather than the mode or mean, because of delayed feedback. During the user elicitation step, we noticed that when a user decides to change the position of the dial, it is because he has just heard a sequence of notes that he considers to be more (or less) pleasing than those he heard previously. Thus, the label indicates the preference of the past sequence. The labels are linearly scaled down
to the range \([-1, 1]\). Since the data in \(\tilde{D}\) is small and unbalanced, we use stratified sampling over solos to divide the dataset into training and validation sets. We then use bagging to create an ensemble of five models for the final estimate.

**Network Architecture**

We estimate the function \(g_{\phi}\) using transfer learning from BebopNet. The user preference model consists of the same layers as BebopNet without the final fully-connected layers. Next, we apply scaled dot-product attention [VSP\textsuperscript{+}17] over \(\tau\) time steps followed by fully-connected and tanh layers. The transferred layers are initialized using the weights \(\theta\) of BebopNet. Furthermore, the weights of the embedding layers are frozen during training.

**Selective Prediction**

To elevate the accuracy of \(g_{\phi}\), we utilize selective prediction whereby we ignore predictions whose confidence is too low. We use the prediction magnitude as a proxy for confidence. Given confidence threshold parameters, \(\beta_1 < 0, \beta_2 > 0\), we define \(g_{\phi, \beta_1, \beta_2}(X_i^t)\) in Eq. 2.3.

\[
g_{\phi, \beta_1, \beta_2}(X_i^t) = \begin{cases} 0 & \text{if } \beta_1 < g_{\phi}(X_i^t) < \beta_2 \\ g_{\phi}(X_i^t) & \text{else} \end{cases} \tag{2.3}
\]

The parameters \(\beta_1\) and \(\beta_2\) change our coverage rate and are determined by minimizing error (risk) on the risk-coverage plot along a predefined coverage contour. More details are given in Section 2.5.2.

**2.4.4 Optimized Music Generation**

To optimize generations from \(f_{\theta}\), we apply a variant of beam-search, \(\psi\), whose objective scores are obtained from non-rejected predictions of \(g_{\phi}\). Pseudocode of the \(\psi\) procedure is presented in the appendix (Section 1). We denote by \(V_b = [X_1^t, X_2^t, ..., X_b^t]\) a running batch (beam) of size (beam-width) \(b\) containing the most promising candidate sequences found so far by the algorithm. The sequences are all initialized with the starting input sequence. In our case, this is the melody of the jazz standard. At every time step \(t\), we produce a probability distribution of the next note of every sequence in \(V_b\) by passing the \(b\) sequences through the network \(f_{\theta}(X_i^t, c_{i+1}^t)\). As opposed to typical applications of beam-search, rather than choosing the most probable notes from \(Pr(s_{i+1}|X_i^t, c_{i+1}^t)\), we independently and randomly sample from the output distribution. This encourages diversity in our considered options. We then calculate the score of the extended candidates using the preference metric, \(g_{\phi}\).

Every \(\delta\) steps, we perform a beam update process. We choose the highest scoring \(k\) sequences calculated by \(g_{\phi}\). Then we duplicate these sequences \(b/k\) times to main-
tain a full beam of $b$ sequences. Choosing different values of $\delta$ allows us to control a horizon parameter, which facilitates longer term predictions when extending candidate sequences in the beam. The use of larger horizons may lead to sub-optimal optimization but increases variability.

### 2.5 Experiments

We start the experimental process by training BebopNet as described in Section 2.4. After training, we use BebopNet to generate multiple solos over different jazz standards\(^5\).

To verify that BebopNet can generalize to harmonic progressions of different musical genres, we also generate improvisations over pop songs (see the appendix, Section 7.2).

This section has two sub-sections. First, we evaluate BebopNet in terms of harmonic coherence (2.5.1). Next, we present an analysis of our personalization process (2.5.2). All experiments were performed on desktop computers with a single Titan X GPU. Hyperparameters are provided in the appendix Section 7.4.1.

#### 2.5.1 Harmonic Coherence

We begin by evaluating the extent to which BebopNet was able to capture the context of chords, which we term harmonic coherence. We define two harmonic coherence metrics using either scale match or chord match. These metrics are defined as the percent of time within a measure where notes match pitches of the scale or the chord being played, respectively. We rely on a standard definition of matching scales to chords using the chord-scale system [CHC02]. While most notes in a solo should be harmonically coherent, some non-coherent notes are often incorporated. Common examples of their uses are chromatic lines, approach notes and enclosures [Coc91]. Therefore, as we do not expect a perfect harmonic match according to pure music rules, we take as a baseline the average matching statistics of these quantities for each jazz artist in our dataset. The harmonic coherence statistics of BebopNet are computed over the dataset used for the preference metric learning (generated by BebopNet), which also includes chord progressions not heard during the jazz modeling stage. The baselines and results are reported in Table 2.1. It is evident that our model exhibits harmonic coherence in the ‘ballpark’ of the jazz artists even on chord progressions not previously heard.

#### 2.5.2 Analyzing Personalized Models

We applied the proposed pipeline to generate personalized models for each of the four users, all amateur jazz musicians. All users listened to the same training dataset of solos to create their personal metric (see Section 2.4). Each user provided continuous

\(^5\)To appreciate the diversity of BebopNet, listen to seven solos generated for user-4 for the tune Recorda-Me in the appendix (Section 7.2).
<table>
<thead>
<tr>
<th>Name</th>
<th>Adderley</th>
<th>Gordon</th>
<th>Getz</th>
<th>Parker</th>
<th>Rollins</th>
<th>Stitt</th>
<th>Woods</th>
<th>Ammons</th>
<th>BN (Heard)</th>
<th>BN (Unheard)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chord</td>
<td>0.50</td>
<td>0.54</td>
<td>0.53</td>
<td>0.52</td>
<td>0.52</td>
<td>0.53</td>
<td>0.50</td>
<td>0.54</td>
<td>0.53</td>
<td>0.52</td>
</tr>
<tr>
<td>Scale</td>
<td>0.78</td>
<td>0.83</td>
<td>0.81</td>
<td>0.80</td>
<td>0.81</td>
<td>0.83</td>
<td>0.78</td>
<td>0.83</td>
<td>0.82</td>
<td>0.81</td>
</tr>
</tbody>
</table>

Table 2.1: Harmonic coherence: The average chord and scale matches computed for artists in the dataset and for BebopNet (BN). A higher number indicates a high coherency level. BebopNet (BN) is measured separately for harmonic progressions heard and not heard in the training dataset.

feedback for each solo using our CRDI variant. In this section, we describe our evaluation process for user-1. The evaluation results for the rest of the users are presented in the appendix in Section 7.4.3.

We analyze the quality of our preference metric function $g_\phi$ by plotting a histogram of the network’s predictions applied on a validation set. Consider Figure 2.3i. We can crudely divide the histogram into three areas: the right-hand side region corresponds to mostly positive sequences predicted with high accuracy; the center region corresponds to high confusion between positive and negative; and the left one, to mostly negative sequences predicted with some confusion. While the overall error of the preference model is high (0.4 MSE where the regression domain is [-1,1]), it is still useful since we are interested in its predictions in the positive (green) spectrum for the forthcoming optimization stage. While trading-off coverage, we increase prediction accuracy using selective prediction by allowing our classifier to abstain when it is not sufficiently confident. To this end, we ignore predictions whose magnitude is between two rejection thresholds (see Section 2.4.3). Based on preliminary observations, we fix the rejection thresholds to maintain 25% coverage over the validation set. In Figure 2.3ii we present a risk-coverage plot for user-1 (see definition in [GEY17]). The risk surface is computed by moving two thresholds $\beta_1$ and $\beta_2$ across the histogram in Figure 2.3i, and at each point, for data not between the thresholds, we calculate the risk (error of classification to three categories: positive, neutral and negative) and the coverage (percent of data maintained).

We increase the diversity of generated samples by taking the score’s sign rather than the exact score predicted by the preference model $g_\phi$. Therefore, different positive samples are given equal score. For user-1, the average score predicted by $g_\phi$ for generated solos of BebopNet is 0.07. As we introduce beam-search and increase the beam width, the performance increases up to an optimal point from which it decreases (see the appendix, Section 7.4.3). User-1’s scores peaked at 0.8 with $b = 32, k = 8$. Anecdotally, there was one solo that user-1 felt was exceptionally good. For that solo, the model predicted the perfect score of 1. This indicates that the use of beam-search is indeed beneficial for optimizing the preference metric.
Figure 2.3: 2.3i Predictions of the preference model on sequences from a validation set. Green: sequences labeled with a positive score \((y_\tau > 0)\); yellow: neutral \((y_\tau = 0)\); red: negative \((y_\tau < 0)\). The blue vertical lines indicate thresholds \(\beta_1, \beta_2\) used for selective prediction. 2.3ii Risk-coverage plot for the predictions of the preference model. \(\beta_1, \beta_2\) (green lines) are defined to be the thresholds that yield a minimum error on the contour of 25% coverage.

### 2.6 Plagiarism Analysis

One major concern is the extent to which BebopNet plagiarizes. In our calculations, two sequences that are identical up to transposition are considered the same. To quantify plagiarism in a solo with respect to a set of source solos, we measure the percentage of n-grams in that solo that also appear in any other solo in the source. These statistics are also applied to any artist in our dataset to form a baseline for the typical amount of copying exhibited by humans.

Another plagiarism measurement we define is the largest common sub-sequence. For each solo, we consider the solos of other artists as the source set. Then, we average the results per artist. Also, for every artist, we compare every solo against the rest of his solos to measure self-plagiarism. For BebopNet, we quantify the plagiarism level with respect to the entire corpus. The average plagiarism level of BebopNet is 3.8. Interestingly, this value lies within the human plagiarism range found in the dataset. This indicates that BebopNet can be accused of plagiarism as much as some of the famous jazz giants. We present the extended results in the appendix.
Chapter 3
Musical Features

3.1 Introduction
While the results described in Chapter 2 indicate that our proposed pipeline enables personalized improvisation generation, it raises questions regarding the musical features learned and expressed by the user preference metric. These issues drove us towards researching a more explainable model and developing a method for controlling the specific musical features of a generated improvisation. In this chapter, we propose a new approach for generating jazz improvisations by optimizing a set of chosen musical features. We define 15 different musical features that can be set as target features for the generated improvisations. The features were chosen taking into consideration the unique elements of jazz improvisations and, specifically, saxophone bebop improvisations. The features were defined based on a well-known jazz book [Coc91] and music theory principles, after consulting numerous amateur and professional jazz musicians. The following set of musical features allows us to optimize generated improvisations by BebopNet as well as analyze existing jazz improvisations. In the next chapters, we will also utilize the generated feature space for user preference elicitation with active learning, and model training with reinforcement learning. Here, we start by describing the different features. Thereafter, we present a feature-based analysis of our dataset and then present feature-guided improvisation generation with BebopNet.

3.2 Musical Feature List
In this section we define the different target musical features. To balance the different features easily, we normalize each feature to be between 0 and 1. The features are calculated given a whole measure.

Denote the $N_i$ notes existing in a certain measure $m_i$ as $n_0, n_1, ..., n_{N_i}$. $C_i$ denotes the chord used in measure $m_i$. We mark our feature vector as $f$, when feature $j$ is marked as $f_j$. We denote the four beats in a measure as $b_1, b_2, b_3, b_4$. 
3.2.1 Number of notes

We measure the number of notes in a certain measure and normalize it by dividing by 32, the maximum number of notes that BebopNet is able to produce in a measure.

\[ f_1(m_i) = \frac{N_i}{32} \]

3.2.2 Duration entropy

We measure the diversity of the durations used in a certain measure by calculating the entropy of the posterior probabilities of the durations. We expect low entropy when a bias in the distribution exists and high entropy for an even distribution.

Let \( d \) be the set of \( N_d \) different note durations that appear in a measure \( m_i \).

\[ f_2(m_i) = \sum_{j=0}^{N_d} p(d_j) \log p(d_j) \]

3.2.3 Pentatonic scale

The pentatonic scale is commonly used in many musical genres, and specifically jazz and bebop. We calculate the percentage of notes in the measure that belong to the pentatonic scale built on the root of the chord in that measure. The major pentatonic scale consists of the 1st, 2nd, 3rd, 5th and 6th degrees of the scale of the current major chord. The minor pentatonic scale consists of the 1st, 3rd, 4th, 5th and 7th degrees of the scale of the current minor chord.

\[
1_{\text{Pentatonic Scale}}(C_i)(n_j) = \begin{cases} 
1, & n_j \in \text{Pentatonic Scale}(C_i) \\
0, & n_j \notin \text{Pentatonic Scale}(C_i) 
\end{cases}
\]

\[ f_3(m_i) = \frac{\sum_{j=0}^{N_i} 1_{\text{Pentatonic Scale}}(C_i)(n_j)}{N_i} \]

3.2.4 Blues scale

The blues scale, similarly to the pentatonic scale, consists of the five notes of the pentatonic scale, with the additional “blue note”. For a major scale, the blue note is the elevated 2nd degree of the scale. For a minor scale, the blue note is the elevated 3rd degree of the scale. We calculate the percentage of notes in the measure that belong to the blues scale built on the root of the chord in that measure.

\[
1_{\text{Blues Scale}}(C_i)(n_j) = \begin{cases} 
1, & n_j \in \text{Blues Scale}(C_i) \\
0, & n_j \notin \text{Blues Scale}(C_i) 
\end{cases}
\]

\[ f_4(m_i) = \frac{\sum_{j=0}^{N_i} 1_{\text{Blues Scale}}(C_i)(n_j)}{N_i} \]
3.2.5 Syncopation

Syncopation is a common property of bebop jazz. Syncopation is related to rhythm, and occurs when there is a shift of rhythmic accents from the beat. We measure syncopation by inspecting the four down-beats in a measure. We consider a beat as syncopated if there is no note on the down-beat and the next note starts before the next down-beat.

Denote a function $Syncopated(b_i)$ that returns 1 if a beat is considered syncopated.

$$1_{Syncopation}(b_j) = \begin{cases} 
1, & Syncopated(b_j) \\
0, & \text{else} 
\end{cases}$$

$$f_5(m_i) = \sum_{j=1}^{4} 1_{Syncopation}(b_j)$$

3.2.6 Chord note on a downbeat

A common property of jazz improvisations is the usage of chord notes on the four downbeats of the measure. We consider the four beats in the measure, and calculate the number of times the chord notes appear in these beats.

$$1_{Chord\_Note\_Down\_Beat(C_i)}(n_j) = \begin{cases} 
1, & n_j \in b_1, b_2, b_3, b_4 \ \& \ n_j \in C_i \\
0, & \text{else} 
\end{cases}$$

$$f_6(m_i) = \frac{\sum_{j=0}^{N_i} 1_{Chord\_Note\_Down\_Beat(C_i)}(n_j)}{4}$$

3.2.7 Chromatic before a chord note on a down-beat

Extending the previous feature, a common practice is to add “approach notes” before chord notes on downbeats. We examine the previous feature, and further analyze the note before each beat. Our definition for an “approach note” is a difference of one step in pitch (equal to a semi-tone).

$$1_{Chromatic\_Before(C_i)}(n_j) = \begin{cases} 
1, & n_j \in b_1, b_2, b_3, b_4 \ \& \ n_j \in C_i \ \& \ |n_j - n_{j-1}| = 1 \\
0, & \text{else} 
\end{cases}$$

$$f_7(m_i) = \frac{\sum_{j=0}^{N_i} 1_{Chromatic\_Before(C_i)}(n_j)}{N_i}$$

3.2.8 Chord match

We measure harmonic coherence by calculating the percentage of notes that belong to the current chord (and next, the current scale). Since jazz consists of tensions and
releases, we do not expect a jazz improvisation to always include chord notes. Thus, a large score for chord match does not indicate how good the improvisation actually is. We can, however, utilize this score to determine a certain desired level of harmonic coherence in the jazz improvisation.

\[
1_{\text{Chord\_Match}(C_i)}(n_j) = \begin{cases} 
1, & n_j \in C_i \\
0, & n_j \notin C_i 
\end{cases}
\]

\[
f_{9}(m_i) = \frac{\sum_{j=0}^{N_i} 1_{\text{Chord\_Match}(C_i)}(n_j)}{N_i}
\]

### 3.2.9 Scale match

We continue with the second harmonic coherence measurement, and calculate the percentage of notes in the measure that belong to the scale built on the root of the current chord.

\[
1_{\text{Scale\_Match}(C_i)}(n_j) = \begin{cases} 
1, & n_j \in \text{Scale}(C_i) \\
0, & n_j \notin \text{Scale}(C_i) 
\end{cases}
\]

\[
f_{9}(m_i) = \frac{\sum_{j=0}^{N_i} 1_{\text{Scale\_Match}(C_i)}(n_j)}{N_i}
\]

### 3.2.10 Altered scale match

The altered scale may also be used in jazz improvisations. Taking into consideration the scale built on the root of the current chord, the altered scale keeps the basic degrees of the chord (1st, 3rd, 7th), while altering the rest of the degrees by a semi-tone up or down. We measure the percentage of notes that belong to the altered scale in a certain measure.

\[
1_{\text{Altered\_Scale\_Match}(C_i)}(n_j) = \begin{cases} 
1, & n_j \in \text{Altered\_Scale}(C_i) \\
0, & n_j \notin \text{Altered\_Scale}(C_i) 
\end{cases}
\]

\[
f_{10}(m_i) = \frac{\sum_{j=0}^{N_i} 1_{\text{Altered\_Scale\_Match}(C_i)}(n_j)}{N_i}
\]

### 3.2.11 Tension

To construct a jazz improvisation, the musicians join together tensions and releases. The leading principle here is to use a note that does not belong to the current harmony to create tension and, later on, resolve it by using a note that is harmonically coherent. We define tension by using the 9th, 11th, or 13th degree of the scale on a downbeat.
\[ 1_{\text{Tension}}(n_j) = \begin{cases} 1, & n_j \in 9_{th}, 11_{th}, 13_{th} \text{ degree of Scale}(C_i) \\ 0, & \text{else} \end{cases} \]

\[ f_{11}(m_i) = \frac{\sum_{j=0}^{N_i} 1_{\text{Tension}}(n_j)}{N_i} \]

### 3.2.12 Intervals

This feature inspects the melodic line of the improvisation. We inspect the pitch interval between any two consecutive notes in the measure. We then calculate the percentage of intervals in a certain measure that are smaller than three steps (equals to three semi-tones).

\[ 1_{\text{Intervals}}(n_j) = \begin{cases} 1, & |n_j - n_{j-1}| < 3 \\ 0, & \text{else} \end{cases} \]

\[ f_{12}(m_i) = \frac{\sum_{j=1}^{N_i} 1_{\text{Intervals}}(n_j)}{N_i} \]

### 3.2.13 Duration repetition within a whole measure

Repetition is a key property of any musical piece. We first examine the repetition of durations in two consecutive measures. We calculate the number of common durations between the two measures, and divide by the number of notes in the current measure. Note that we do not take the order of notes into account.

\[ f_{13}(m_i) = \frac{|\cap (\text{durations}(m_i), \text{durations}(m_{i-1}))|}{N_i} \]

Also, observe that the absolute value indicates the number of elements in the given set.

### 3.2.14 Duration repetition within a half measure

This feature examines duration repetition in a single measure by dividing it so that there are two sections of two beats each and then compare these as we did for the previous feature.

Denote the two parts of a measure \( m_i \) divided equally as \( m_{iA}, m_{iB} \). \( N_{iA} \) indicates the number of notes in the first part of measure \( m_i \).

\[ f_{14}(m_i) = \frac{|\cap (\text{durations}(m_{iA}), \text{durations}(m_{iB}))|}{N_{iA}} \]

Note that the absolute value indicates the number of elements in the given set.
3.2.15 Pitch repetition

Similarly to duration repetition, we examine pitch repetition in two consecutive measures. We divide the number of repeating pitches by the number of notes in the current measure. Note that we do not take the order of notes into account.

\[ f_{15}(m_i) = \frac{|\cap \{pitches(m_i), pitches(m_{i-1})\}|}{N_i} \]

Observe that the absolute value indicates the number of elements in the given set.

3.3 Dataset Feature Analysis

In this section, we present a musical feature analysis of our dataset (Section 2.4.1). Figure 3.3 presents a histogram of the computed features over our dataset. Each feature is displayed in a different color. The curved lines present an estimation of the distribution of the specific feature in the dataset.

3.4 Feature-Guided Jazz Improvisation Generation

Using the defined features, we employ beam-search to optimize the BebopNet generated improvisations. We start by defining a target feature vector \( t_f \) for BebopNet’s optimisation process. At each step, BebopNet generates multiple options for the next
Table 3.1: Average score for each musical feature. Baseline represents the average score of a feature over BebopNet’s training set. Optimized represents the feature scores of optimized generated improvisations (with a target of a single feature).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Baseline</th>
<th>Optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of notes</td>
<td>0.225</td>
<td>0.510</td>
</tr>
<tr>
<td>Duration entropy</td>
<td>0.520</td>
<td>0.997</td>
</tr>
<tr>
<td>Pentatonic scale</td>
<td>0.593</td>
<td>0.910</td>
</tr>
<tr>
<td>Blues scale</td>
<td>0.626</td>
<td>0.935</td>
</tr>
<tr>
<td>Syncopation</td>
<td>0.070</td>
<td>0.515</td>
</tr>
<tr>
<td>Chord note on a downbeat</td>
<td>0.378</td>
<td>0.665</td>
</tr>
<tr>
<td>Chromatic before a chord note on a down-beat</td>
<td>0.073</td>
<td>0.230</td>
</tr>
<tr>
<td>Chord match</td>
<td>0.443</td>
<td>0.770</td>
</tr>
<tr>
<td>Scale match</td>
<td>0.501</td>
<td>0.910</td>
</tr>
<tr>
<td>Altered scale match</td>
<td>0.206</td>
<td>0.370</td>
</tr>
<tr>
<td>Tension</td>
<td>0.160</td>
<td>0.530</td>
</tr>
<tr>
<td>Intervals</td>
<td>0.546</td>
<td>0.985</td>
</tr>
<tr>
<td>Duration repetition within a whole measure</td>
<td>0.224</td>
<td>0.905</td>
</tr>
<tr>
<td>Duration repetition within a half measure</td>
<td>0.180</td>
<td>0.950</td>
</tr>
<tr>
<td>Pitch repetition</td>
<td>0.266</td>
<td>0.885</td>
</tr>
</tbody>
</table>

measure. For each generated measure \( m_i \), we extract the feature vector \( f(m_i) \). We then calculate a score \( s_i \) for each option by multiplying the target feature vector \( t_f \) by the calculated features \( f(m_i) \).

\[
s_i = t_f \cdot f(m_i)
\]

We then select the options that yield the highest score and continue the search as described in Section 2.4.4.

To numerically show that the optimized generations do fit the desired features, we analyze single feature scores. As a baseline, we calculate the features in our training set (see Section 2.4.1). We generate multiple samples with a single feature as target, and calculate the average resulting score. Table 3.4 shows that the optimized improvisations managed to emphasize the selected feature and achieved a considerably higher score.

Multiple samples of feature-guided jazz improvisations generated by BebopNet can be found at: https://shunithaviv.github.io/feature-guided-bebopnet/
Chapter 4

Active Learning

4.1 Introduction

Training a model with data that requires human labels raises many challenges. A major one is the long and expensive labeling time required to collect enough data that will be sufficient for training, as well as diverse enough to cover the learned space. In this chapter, we attempt to improve the user preference elicitation process (Section 2.4.2). This process includes presenting jazz improvisations to a certain user and collecting their preference feedback. Thereafter, we use the collected labels to train the user preference metric to predict the user preference.

While in Section 2.4.2 we did not use any special method to choose the order of the samples for labeling (passive learning), here, we examine the influence of labeling data using active learning. Moreover, we examine a different labeling task that utilizes our musical feature analysis and feature-guided generation (presented in Chapter 3).

4.2 Related work

Active learning can be applied to scenarios in which an information source is required to label data. In these scenarios, where labeling may be expensive and demanding, labeling a complete dataset may be redundant, inefficient and even completely out of reach. Active learning is designed to interactively query the information source for labels, choosing the most informative queries to benefit the learning process. Thus, the number of labeled samples required for the learning task may be significantly lower than with passive learning.

Active learning strategies include uncertainty sampling [TK01, JPP09, Set09, YMN+15], query by committee [GBNT05, IKM+11], error minimization [HJZL06, JPP09] and more. In our work we focus on uncertainty sampling, in which the active learner chooses to query the points about which the current learner is the most uncertain.
4.3 Methods

We train our active learner to predict a user preference given a jazz improvisation of four measures, with the general rule that the higher the prediction, the better the preference, and vice versa.

We gathered a pool of four measure improvisations played along with common jazz chord progressions. To create a large variety of improvisation, we utilize the feature-guided generation with different target feature distributions. After generating the improvisations, we can map them to the feature space, so that each improvisation is represented by a single feature vector, which is the mean of the four feature vectors of the four measures. Note that this feature vector is computed after the generation, in contrast to the target feature vector that guided the generation.

Our active learner includes a neural network \( a(f^i) \) that predicts a score \( s^i \) given a vector of musical features \( f^i \). \( a(f^i) \) consists of two fully connected layers with ReLU activation functions and a final fully connected layer. We aim for the score to express the user preference. Since defining an absolute number for musical preference is challenging, we ask our users to listen to two improvisations and choose their preferred one. This comparison method was found to be easier for the users to label with consistency. For the active learner to be able to work with this labeling method, we use Siamese networks [BVH+16] to train our learner. At each step of the training process, given two improvisations with feature vectors \( f^1, f^2 \), we use the neural network to predict the scores \( s^1, s^2 \). Then, our loss function is \( \text{sigmoid}(s^1 - s^2) = l \), when \( l \) indicates the user label: \( l = 1 \) if the user preferred the first improvisation over the second, and \( l = 0 \) if the user preferred the second. Therefore, our neural network is trained to predict a higher number for preferred improvisations. Our active learner works interactively. At each step, the learner queries the user with a pair of improvisations using uncertainty sampling. To choose the next candidate, the active learner predicts the scores for all improvisations and calculates the sigmoid of the score differences for each pair. A result close to 0.5 indicates high uncertainty for a certain pair, meaning the learner cannot predict with high certainty which improvisation will be preferred by the user. The pair with a result closest to 0.5 will be chosen next for the user to label. To initialize our learner at the beginning of the process, we choose random pairs for training.

4.4 Experiments and Results

In this section we describe the experiments conducted to test our active learner. We start by defining a rule to imitate a musical preference, and later experiment with a human labeled dataset of pairs of improvisations.
4.4.1 Active Learning vs. Passive Learning: Predefined Target Feature Vector

Here, we measure the improvement achieved by using active learning over passive learning. To focus on the learning algorithm and set aside noise that results from human labeling, we define a preference rule using a target feature vector. While using automatic labeling may indicate the algorithmic benefits of active learning, it differs substantially from human annotations which do not usually adhere to simple rules and are affected by environmental and emotional factors.

We generate a set of 184 jazz improvisations, using a variety of target musical feature vectors. We then assemble a dataset of 33856 pairs of the aforementioned improvisations. Using the predefined target feature vector, we automatically label our dataset of pairs, so that the improvisation with features closest to our target will be preferred.

We simulate a learning process with our active learner as well as with a passive learner that randomizes a pair to be queried at each step. We separate a validation set to test the accuracy of our learners as the learning process progresses.

Figure 4.1 presents an experiment with a one-hot target feature vector $f^{*2}$ (equation 4.1). We simulated the learning process for both the active learner and the passive learner. There is a clear advantage for the active learner in terms of validation accuracy and convergence rate. Figure 4.2 presents an experiment with a five-hot target feature vector $f^{*2}$ (equation 4.2). Although there is an advantage to the active learner, it is less significant. The higher benefit for active learning with one-hot feature vector may be due to the relatively small feature set. A more complex decision rule may take more steps for the algorithm to learn, resulting in a less significant advantage for active learning, especially with such a small training set.

$$f^{*1} = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0] \quad (4.1)$$

$$f^{*2} = [1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0] \quad (4.2)$$

4.4.2 Active Learning vs. Passive Learning: Human Labeled Dataset

We now test our active learner with human labels. In contrast to the automatically labeled dataset, human labeled datasets can be inconsistent and noisy, thus complicating the learning process. We generated 91 jazz improvisations to assemble a dataset of 811 pairs. The dataset of pairs was manually labeled by a human annotator according to their musical preference.

We test our learners by comparing accuracy on a validation set. Since our training set is relatively small and we wish to use as much of it as we can for training, we use reverse cross-validation for testing. We divide the training set into three parts, training
Figure 4.1: Active Learning vs. Passive Learning convergence checks: 1-hot feature target preferred. The graphs are an average of 10 runs, and the confidence interval is for 95%.

three times with each of the parts as a training set and the rest as a validation set. Then, we average the results. To set a baseline, we trained our learner with all labeled pairs to achieve an accuracy of 0.75.

We simulate a training process with both the active learner and the passive learner. Figure 4.3 displays the results. With human labels, active learning does not have an advantage in the learning process.

4.5 Conclusions

While active learning shows a clear advantage in learning when an automated user preference system is used, it does not improve the process when a human labeled dataset is available. Human labels may be extremely noisy and inconsistent, and they are subject to multiple environmental factors as well as mood and musical background. Moreover, user preference may differ in the process of labeling. A central effect of labeling pairs constructed out of a small set was the increasing repetition and familiarity of the improvisations, causing the labeling process to be tiring for the user while also directly affecting the results of our experiments. To tackle these challenges, future work may include consistency analysis of the user labels and using a larger dataset to prevent repetition in the labeling process.
Figure 4.2: Active Learning vs. Passive Learning convergence checks: 5-hot feature target preferred. The graphs are an average of 10 runs, and the confidence interval is for 95%.

Figure 4.3: Active Learning vs. Passive Learning convergence checks: Manually labeled dataset
Chapter 5

Reinforcement Learning

5.1 Introduction

In this chapter, we explore an alternative approach for controlling the generation of jazz improvisations. We research the benefits of employing Reinforcement Learning (RL) to train a new optimized generative model. To this end, we set our reward function to be either a user’s personal preference or a musical feature setting.

In Section 5.3, we present our reinforcement learning model for jazz improvisations which is based on the actor-critic method (see Section 5.2.2). Given the previous sequence of notes, we set BebopNet as the actor, determining the policy that predicts the note to be played next. Our critic aims to evaluate the policy by predicting the value, which indicates the quality of the currently generated note sequence (as driven by the defined reward). We train the model by altering between policy updates (actor) and value prediction updates (critic). Finally, the trained policy is used as our optimized generative model. For this task, we created a new RL environment for jazz improvisations (see Section 5.3.1).

We first evaluate our RL model on a fixed reward, inspired by musical features. This task’s goal is to see if the trained RL model can generate musical sequences that match the musical feature based reward. Section 5.4 describes our experiments and shows our results on different musical feature based rewards.

Next, we replace the reward function with a user preference metric (as described in Section 2.4.3). Here, the trained RL model adapts its policy to match the user preference as expressed by the reward and thus generates personalized jazz improvisations. We present our results in Section 5.4.1. Finally, we discuss the challenges of using RL with a human preference based reward, and offer ideas for future research.

5.2 Related Work

RL is not commonly involved in generative models for music. A few applications for text generation, however, do include two such approaches. The first approach employs
MIXER (Mixed Incremental Cross-Entropy Reinforce) [RCAZ15], a text translation model that uses BLEU (Bilingual Evaluation Understudy Score) [PRWZ02] as a reward signal to gradually introduce an RL loss. Initially, the model is pre-trained using cross-entropy; later the training process is repeated with the cross-entropy loss combined with RL. The second approach applies an actor-critic method [BBX+16]. The actor is initialized with the policy of a recurrent neural (RNN) network pre-trained with next step prediction. The critic is trained using the BLEU score to output the value of each word. Both approaches assume the complete task reward is available, and since training with RL from scratch is difficult, they use pre-training to form a basic generative policy using supervised learning before using RL to augment the learning process. When trying to adapt these kind of models to music generation, we must consider the lack of a well-defined reward function. Thus, the musical sequence prediction must be learned mostly from the data itself rather than from the reward.

Another approach that applies RL to an RNN is SeqGan [YZWY17]. This method uses a discriminator network – similarly to Generative Adversarial Networks (GANs) [GPAM+14] – to classify the realism of a generated sequence. There are a few applications for this approach designed for music generation [Hus15, Mog16].

Google’s Magenta group [JGTE16] propose a solution in the symbolic domain of musical notes for harmony-dependent monophonic melody generation. RL is combined to refine the generated musical sequences by applying a pre-defined musical reward function that is based on music theory rules. The chosen RL model is Double DQN (deep Q-Learning; see Section 5.2.1 for a detailed description of the model). The main novelty in that paper is the use of RL for music composition. To date, generative models for language or music consisted of neural networks in the form of some recurrent neural networks, mostly LSTM. Magenta’s new model exploits the advantages of such a system and overcomes the downsides by using RL. The training process initiates with training Note RNN in order to learn how to make note-to-note prediction from data. The trained weights from the Note RNN supply the initial weights for both the DQN neural networks and the Reward RNN, that is held fixed and predicts the reward based on the trained Note RNN. The authors use musical qualities defined on page 42 of Gauldin’s book [Gre89] to define the musical reward $r_{MT}(a, s)$ to the generated melodies. For the model to be creative and use the knowledge it gained when learning next note prediction from data, there is an addition to the reward $\log p(a|s)$, the log probability of a note given a composition $s$. This is incorporated it into the reward function. The total reward given at time $t$, is therefore,

$$r(s, a) = \log p(a|s) + r_{MT}(a, s)/c$$

$c$ is a constant, controlling the emphasis placed on the music theory reward, and is designated to weight and scale the different reward components. Plugging in this
reward to the DQN equation, the new loss function becomes:

\[ L(\theta) = E_\beta[(\log p(a|s) + r_{MT}(a,s)/c + \gamma \max Q(s', a'; \theta) - Q(s, a; \theta))^2] \]

\[ \pi_0(a|s) = \delta(a = \arg \max Q(s, a; \theta)) \]

To train and test the suggested model, the authors define a few constraints on the generated compositions and data representation. Melodies are quantized at the granularity of a sixteenth note, so each time step corresponds to one sixteenth of a bar of music. The range of notes is limited to three octaves, and the musical scale used was limited to C major only.

5.2.1 Deep Q-Learning

Q-learning is a model-free RL algorithm. In each step \( t \), given the state of the environment \( x_t \), the agent takes an action \( u_t \) according to a policy \( \mu_\theta(u_t|x_t) \), receives a reward \( r(x_t, u_t) \), and the environment transitions to the new state \( x_{t+1} \). The goal is to find the optimal \( \theta \) that maximizes the reward, and is known to satisfy the Bellman optimality equation for Q-learning:

\[ Q^*(x, u) = r(x, u) + \gamma \sum_{y'} P(y, u'|x, u) \max \limits_{u'} Q^*(y, u') \]

where \( Q^*(x, u) = \max \limits_{\theta} E[B|x_0 = x, u_0 = u], \forall x \in X, u \in U \)

Deep Q-learning (DQN) uses a neural network to approximate the Q function. The network parameters \( \theta \) are learned by applying stochastic gradient descent (SGD) updates with respect to the TD\( ^2 \) loss function:

\[ L(\theta) = (r(x_t, u_t) + \gamma Q_{\text{target}}(x_{t+1}, u_{t+1}) - Q_\theta(x_t, u_t))^2 \]

A common improvement to DQN is Double DQN, which uses two deep neural networks, one for approximating the Q function, and the other as a target network that is updated less frequently to keep stability.

5.2.2 Policy Gradient – A2C

Actor-critic methods implement a generalized policy iteration – altering between a policy evaluation that is performed by a process called critic, and a policy improvement step performed by a process called actor. This method is both value-based and policy-based, meaning the critic approximates the value function and the actor estimates the policy \( \pi \).

Advantage actor critic (A2C) uses the advantage function to define the policy gradient:

\[ \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a, s) A^\pi(a, s)] \]
At each iteration, the actor collects a set of trajectories by executing the current policy. At each timestep \( t \) in each trajectory, we compute the return \( R_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'} \) and the advantage estimate with respect to the baseline \( \hat{b} \): \( \hat{A}_t = R_t - b(s_t) \), and re-fit the baseline by minimizing \( ||b(s_t) - R_t||^2 \). Finally, the policy is updated, using a policy gradient estimate \( \hat{g} \) that is the sum of all terms \( \nabla_{\theta} \log \pi(a_t, s_t, \theta) \hat{A}_t \).

5.3 Reinforcement Learning with BebopNet

Similarly to Magenta [JGTE16], we implement an architecture that combines BebopNet with RL. We chose to implement A2C instead of DQN, mainly because A2C method can be implemented and integrated more easily with our existing model.

We integrated an existing A2C implementation [Kos18] with our model in the same model construction suggested in this paper. We load our pre-trained BebopNet model, and extend it to A2C model that learns to predict the value function and the next action. The extended RL model includes BebopNet as described in 2.4.1 and follows with an additional LSTM layer and multiple linear layers to predict the value and action.

5.3.1 Jazz Environment

To integrate BebopNet with A2C, we create an environment dedicated to our jazz project. Since we desire to train our model with multiple environments in parallel, our environment is implemented efficiently as a vector of environments. Music compositions do not have a typical environment behavior, and thus, our environment stores the notes in the current composition and returns the last note played as an observation. In each step of the environment, the environment adds the new note chosen as the next action, computes the reward and returns whether the composition ended, when it reaches the determined number of musical measures. In addition, we implement a rendering function that enables us to view the musical sheet of the current composition with the use of the Muse-Score program.

5.3.2 Reward Function

We implement a musical reward function that rewards according to music-theory rules (See Section 5.4). For example, if the played note belongs to the current chord or scale, it will receive a positive reward. If not, it will receive a negative reward. In addition, since we want the compositions to relate to the dataset we initially learned from, we add a reward that aims to keep the action distribution relatively close to the pre-trained LSTM. Related work added this kind of reward by incorporating the log probability of a note given a composition \( s \). We extend this idea by adding the KL divergence of the actions probabilities, and thus reward the model to preserve the knowledge pre-learned
from data. The last ingredient of our reward function is a penalty for overly-repeating notes.

5.4 Experiments and Results

We pre-train BebopNet as described in Section 2.4.1. The training results are displayed in Figures 5.1, 5.2, 5.3. After training BebopNet, we load it into the RL model as described in Section 5.3. The complete model is displayed in Figure 5.4.

We tested multiple reward functions, which include the main principles that are described in [JGTE16]. Let us denote the total reward at step $t$ as $R_t$ and the output probabilities of the actor for pitch and duration as $P_{pitch}$, $P_{duration}$.

First, we tested a reward based on the association of the generated note to the harmonic context: the scale and chord.

$$
R_{scale}^t = \begin{cases} 
1, & \text{note}_t \in \text{scale}_t \\
-1, & \text{note}_t \notin \text{scale}_t 
\end{cases}
$$

$$
R_{chord}^t = \begin{cases} 
1, & \text{note}_t \in \text{chord}_t \\
0, & \text{note}_t \notin \text{chord}_t 
\end{cases}
$$

The use of this reward caused the model to play the same note repetitively. To solve this, we added a penalty for repeating notes.

$$
R_{repeating}^t = \begin{cases} 
n, & \text{last n notes had the same pitch, } n > 2 \\
0, & \text{else} 
\end{cases}
$$

In addition, since we desire to keep the diversity and the knowledge of the initial trained LSTM, we added a reward based on the similarity of the output distributions between the RL model and the original LSTM model.

$$
R_{D_{KL}} = D_{KL}(P_{pitch}||Q_{pitch}) + D_{KL}(P_{duration}||Q_{duration})
$$

$$
D_{KL}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}
$$

After training with the above reward and testing a few weighting options, the model converged to play only rest notes, meaning only silence. Thus, we added a penalty for playing rest notes:

$$
R_{rest}^t = \begin{cases} 
1, & \text{note}_t \text{ is rest} \\
0, & \text{else} 
\end{cases}
$$

The total reward is constructed of multiple rewards, when each of them has a
Figure 5.1: The loss of the pre-trained LSTM model (BebopNet for RL)

Figure 5.2: The accuracy of the pitch of the estimated next note (by BebopNet for RL)

Figure 5.3: The accuracy of the duration of the estimated next note (by BebopNet for RL)
Figure 5.4: The complete model architecture we used. (model) contains the pre-trained LSTM, (rnn) is the additional LSTM and (actor_base), (actor_pitch), (actor_dur), (critic) and (critic_linear) are feed-forward layers dedicated to the A2C predictions.

designated coefficient $c$ that is meant to weight and scale the different rewards.

$$R_t = -c_{DKL} \cdot R_t^{DKL} + c_{scale} \cdot R_t^{scale} + c_{chord} \cdot R_t^{chord} - c_{repeating} \cdot R_t^{repeating} - c_{rest} \cdot R_t^{rest}$$

We uploaded some samples that display the difficulty in defining a suitable reward in the symbolic domain of music. Most of the time, the generated samples contained mostly the same note repeatedly, and other times the compositions were not so pleasant to hear. As discussed before, numerical evaluation cannot implicitly indicate if a musical composition is good or not, so we invite you to judge for yourselves here: link.

The compositions were incorporated with backing tracks generated with the Band In A Box program, and the monophonic piano playing is the generated output. Notice that in jazz music there is a lead melody that opens every musical part, and after one loop of that melody, our improviser begins to play.

5.4.1 User Preference Reward

In Chapter 2, we present our personalization pipeline for jazz improvisations. We use beam search to guide the generative model to create improvisations optimized to the user’s personal taste. Alternatively, instead of beam search, RL can be applied to achieve a similar objective. Using a reward function that reflects the user’s musical preference, we can train a model with RL to generate jazz improvisations that satisfy that user’s taste. After training the user preference metric as described in Section 2.4.3,
Figure 5.5: Example of the convergence of the reward during the training process.

we can set it as the reward function for the RL. Thus, the model will be rewarded according to the correctness of its predicted user preference.

We experimented with different settings for training a model with RL and the user preference metric. Unfortunately, the system failed to converge, which resulted in a poor final performance. There may be numerous possible reasons for this behavior. First, RL uses exploration to find the best path to a high reward. Training Bebop-Net on a limited set of bebop improvisations may lead to a limited distribution of the generated improvisation, thus, limiting the RL in the process of exploration. Second, sampling note by note for generating improvisations by BebopNet as the model trains, may result in an out-of-distribution musical sequence. A musical sequence as such may result in a false predicted user preference, which was trained on a limited training set, thus leading to divergence. We attempted to tackle this issue by incorporating KL-divergence between the pre-trained BebopNet and the generative model for RL. Nevertheless, overcoming this issue was not enough for the model to converge. Another central issue is the difficulty in learning human preferences. Not only are human preferences extremely noisy, but they are also subject to context (mood, environment, sound settings, etc.). As discussed in Section 2.5.2, this issue leads to high confusion in the prediction of any trained preference metric, leading to a difficulty in RL convergence due to reward inconsistency.
Future works that will be able to overcome these issues may be able to train a personalized generative model using RL. Extracting a precise user preference without being sensitive to environmental factors may enable the use of RL to create such a personalized generative model. Another interesting future direction will be to combine user feedback as part of the training process with RL. Incorporating active learning with this approach may result in an efficient learner for personal preference.
Chapter 6

Conclusions and open questions

We presented a novel pipeline for generating personalized harmony-constrained jazz improvisations by learning and optimizing a user-specific musical preference model. To distill the noisy human preference models, we used a selective prediction approach. We introduced an objective evaluation method for subjective content and numerically analyzed our proposed pipeline on four users. We presented several musical features to define a novel feature space for jazz improvisations. To generate feature-based improvisations, we utilized our optimization process with the musical features. We introduced an active learner for user preferences and inspected the benefits of using active learning vs. passive learning in different settings. We proposed an alternative method for training a personalized jazz improviser using RL. We discussed our results and the challenges when using RL with limited human labeled datasets.

Our work raises many questions and directions for future research. While our generated solos are locally coherent and often interesting/pleasing, they lack the qualities of professional jazz related to general structure such as motif development and variations. Preliminary models we have trained on smaller datasets were substantially weak. Can a much larger dataset generate a significantly better model? To acquire such a large corpus it might be necessary to abandon the symbolic approach and rely on raw audio.

Our work emphasizes the need to develop effective methodologies and techniques to extract and distill noisy human feedback that will be required for developing many personalized applications. Our proposed method raises many questions. To what extent does our metric express the specifics of one’s musical taste? Can we extract precise properties from this metric? Additionally, our technique relies on a sufficiently large labeled sample to be provided by each user, a substantial effort on the user’s part. We anticipate that the problem of eliciting user feedback will be solved in a completely different manner, for example, by monitoring user satisfaction unobtrusively, e.g., using a camera, EEG, or even direct brain-computer connections.

The challenge of evaluating neural networks that generate art remains a central issue in this research field. An ideal jazz solo should be creative, interesting and meaningful. Nevertheless, when evaluating jazz solos, there are no mathematical definitions for these
properties—as yet. Some of the main properties of creative performance are innovation
and the generations of patterns that reside out-of-the-box— namely, the extrapolation
of outlier patterns beyond the observed distribution. Present machine learning regimes,
however, are mainly capable of handling interpolation tasks and not extrapolation. Is
it at all possible to learn the patterns of outliers?
Chapter 7

Appendix: BebopNet – Personalized Improvisation Generation

This chapter includes the appendix for Chapter 2.

7.1 Implementation on GitHub

We provide an open-source implementation of BebopNet training and the our personalization pipeline. Moreover, we include a pre-trained model for generating jazz improvisations with BebopNet. The code is available in: 
https://github.com/shunithaviv/bebopnet-code

7.2 Music Samples

We provide a variety of MP3 files of generated solos in: 
https://shunithaviv.github.io/bebopnet

Each sample starts with the melody of the jazz standard, followed by an improvisation whose duration is one chorus. Sections BebopNet in sample and BebopNet out of sample contain solos of BebopNet without beam search over chord progressions in and out of the imitation training set, respectively. Section Diversity contains multiple solos over the same standard to demonstrate the diversity of the model for user-4. Section Personalized Improvisations contain solos following the entire personalization pipeline for the four different users. Section Harmony Guided Improvisations contain solos generated with a harmonic coherence score instead of the user preference score, as described in Section 7.4.2. Section Pop songs contains solos over the popular non-jazz song. Some of our favorite improvisations by BebopNet are presented in the first section, Our Favorite Jazz Improvisations.
7.3 Methods

7.3.1 Dataset Details
A list of the solos included in our dataset is included in section 7.5.

7.3.2 Musical Preference Labeling System
A figure of our CRDI variant is presented in Figure 7.1.

Figure 7.1: Digital CRDI controlled by a user to provide continuous preference feedback.

7.3.3 Beam Search
A pseudo code of the beam search procedure is presented in Algorithm 1.

Beam Search - Complexity Analysis
In terms of time complexity, for every time step, we forward the sequences of length \( t \) through the two networks \( f_\theta \) and \( g_\phi \). A naive implementation would amount to \( O(b \cdot t^2) \) time complexity and \( O(b \cdot t) \) space complexity. Using simple bookkeeping, passing the entire sequences through the networks can be avoided, thus reducing time complexity to \( O(b \cdot t) \) as well. We note that in modern frameworks, it is natural to efficiently implement the multiple forwards through the network in parallel as one would forward a batch.

7.4 Experiments

7.4.1 Hyper-parameter search
Hyper-parameters for both models were selected by performing a manual coarse-to-fine search. Table 7.1 displays considered and chosen hyper-parameters for \( f_\theta \). For user preference metric learning, we chose hyper-parameters using five-fold cross-validation over the training set. The five models used for cross-validation were later combined as an ensemble model. We show the considered hyper-parameters in Table 7.2. Table 7.3 presents the hyper-parameters considered for the beam search.

---

1Image credit: https://github.com/Andrew-Shay/python-gauge
Algorithm 1: Score-based beam search $\psi$

**Input:** jazz model $f_\theta$, score model $g_\phi$, batch size $b$, beam size $k$, update interval $\delta$

**Output:** sequence $X_{\tau+T} = x_1 \cdots x_{\tau+T} \in X^{\tau+T}$

$.\quad$ $V_b = \left[ X_{\tau}^{in}, X_{\tau}^{in}, \ldots, X_{\tau}^{in} \right] \in X^{\tau \times b};$
$.\quad$ $b$ times

$.\quad$ scores $= \left[ -1, -1, \ldots, -1 \right] \in \mathbb{R}^b$
$.\quad$ $b$ times

for step $t$ in $\tau, \tau+1, \ldots, \tau+T$ do

for sequence $X^i_t$ in $V_b$ do

$.\quad$ $Pr(s_{t+1} | X^i_t, c^i_{t+1}) = f_\theta(X^i_t, c^i_{t+1});$

$.\quad$ $s^i_{t+1} \sim Pr(s_{t+1} | X^i_t, c^i_{t+1});$

$.\quad$ $x^i_{t+1} = (s^i_{t+1}, c^i_{t+1});$

$.\quad$ $X^i_{t+1} = x^i_1 \cdots x^i_{t+1};$

$.\quad$ scores$[i] = g_\phi(X^i_{t+1})$

end

$.\quad$ $V_b = \left[ X_{t+1}^1, X_{t+1}^2, \ldots, X_{t+1}^b \right] \in X^{(t+1) \times b};$

if $(t - \tau) \mod \delta = 0$ then

$.\quad$ topk_\text{inds} = k$-argmax scores;

$.\quad$ for $i$ in $1, 2, \ldots, b$ do

$.\quad$ $V_b[i] = V_b[\text{topk}_\text{inds}[i \mod k]]$

end

end

$.\quad$ $X_{\tau+T} = \arg\max_X \text{score}(V_b)$

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Lowest considered</th>
<th>Top considered</th>
<th>Chosen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of layers</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Learning rate</td>
<td>$1 e^{-3}$</td>
<td>5</td>
<td>$5 e^{-1}$</td>
</tr>
<tr>
<td>Weight decay</td>
<td>$1 e^{-7}$</td>
<td>$1 e^{-2}$</td>
<td>$1 e^{-6}$</td>
</tr>
<tr>
<td>Dropout</td>
<td>0</td>
<td>0.8</td>
<td>0</td>
</tr>
<tr>
<td>Number of epochs</td>
<td>100</td>
<td>600</td>
<td>500</td>
</tr>
<tr>
<td>Batch size</td>
<td>32</td>
<td>512</td>
<td>256</td>
</tr>
<tr>
<td>Sequence Length</td>
<td>8</td>
<td>150</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 7.1: Hyper-parameter search: considered range and chosen values for note prediction
<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Lowest considered</th>
<th>Top considered</th>
<th>Chosen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pitch embedding size</td>
<td>64</td>
<td>1024</td>
<td>256</td>
</tr>
<tr>
<td>Duration embedding size</td>
<td>64</td>
<td>1024</td>
<td>256</td>
</tr>
<tr>
<td>Hidden size</td>
<td>128</td>
<td>2048</td>
<td>512</td>
</tr>
<tr>
<td>Number of layers</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Learning rate</td>
<td>$1e^{-2}$</td>
<td>$5e^{-2}$</td>
<td>$1e^{-1}$</td>
</tr>
<tr>
<td>Weight decay</td>
<td>$1e^{-7}$</td>
<td>$1e^{-2}$</td>
<td>$1e^{-6}$</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.4</td>
<td>0.8</td>
<td>0.9</td>
</tr>
<tr>
<td>Number of epochs</td>
<td>50</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>Batch size</td>
<td>32</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>Sequence Length</td>
<td>8</td>
<td>32</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 7.2: Hyper-parameter search: considered range and chosen values for user preference metric learning

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Lowest considered</th>
<th>Top considered</th>
<th>Chosen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam width $k$</td>
<td>2</td>
<td>500</td>
<td>32</td>
</tr>
<tr>
<td>Beam depth</td>
<td>1 note</td>
<td>4 measures</td>
<td>2 measures</td>
</tr>
</tbody>
</table>

Table 7.3: Hyper-parameter search: considered range and chosen values for the beam search

### 7.4.2 Harmonic Guided Generation

An alternate known generative approach to the one we propose in our work is to maximize a predefined reward function based on music theory, see, e.g., [JGTE17, Fra06], rather than a user-specific metric. This approach, similar to “reward hacking” common in RL [AOS+16], may lead to undesired results. To examine a simple baseline using this method, we generated samples from BebopNet that are optimized to play notes within the harmony. We then used the harmony coherence metric (for scales), as discussed in Section 2.5.1, and applied beam-search with it. The resulting optimized solos successfully maximized this harmonic coherence metric. One can listen to the generated solos that appear in the music samples in Section 7.2. Perhaps unsurprisingly, the use of this handcrafted metric resulted in a degraded performance where the solos were biased to prefer repeating notes that match the chord.

### 7.4.3 Per-User Personalization

**Elicitation Process**

We applied our proposed pipeline on four users, all of whom are amateur jazz musicians. Each of the users has a few years of experience in jazz improvisation, however, music is not their main profession. For each user, the elicitation process took place in one session of 5 hours in front of a desktop. The users used the computer keyboard arrows to change the CRDI meter while they listened to the improvisations with headphones. Before
<table>
<thead>
<tr>
<th>User</th>
<th>$\beta_1$</th>
<th>$\beta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.679</td>
<td>0.198</td>
</tr>
<tr>
<td>2</td>
<td>-0.999</td>
<td>0.128</td>
</tr>
<tr>
<td>3</td>
<td>-0.515</td>
<td>0.086</td>
</tr>
<tr>
<td>4</td>
<td>-0.865</td>
<td>0.158</td>
</tr>
</tbody>
</table>

Table 7.4: $\beta_1, \beta_2$ for every user. $\beta_1, \beta_2$ are defined to be the thresholds that yield minimum error on the contour of 25% coverage.

playing each improvisation, the name of the standard over which the improvisation is played is shown. The user may choose to play the melody of the standard before listening to the improvisation to familiarize themselves with the standard.

Results

Below we present the analysis for all users. Figures 7.2, 7.3, 7.4, 7.5, present the histograms of predictions over sequences from the validation set for all users. Table 7.4 presents the thresholds selected for selective prediction for each user. Different users indeed exhibit different taste. One such contrast is the first-order statistics of users 1 and 2. While user-1 labeled a large proportion of the data as negative, user-2 labeled most of it as positive. In contrast to the two above, user-3 has a high level of neutral sequences, which may indicate uncertainty in his preference. In Figure 7.6 we can see a similar behavior in all the users’ models $g_{\phi}$: the beam size increases the score obtains grows up to an optimal point. A noticeable improvement of the user preference score is achieved for all the users, as we compare the initial score for BebopNet (when $\text{beamwidth} = 1_1$) to the top score achieved with beam search and the preference model.

7.4.4 Plagiarism Analysis

As described in Plagiarism section, we present here the plagiarism analysis results. Table 7.4.4 presents the average longest sub-sequence between any two artists. The diagonal of this table represents "self-plagiarism". Figure 7.7 displays the percent of identical sequences in length $n$ per artist.
Figure 7.2: User 1: 7.2i Predictions of the preference model on sequences from a validation set. Green: sequences labeled with a positive score ($y_\tau > 0$); yellow: neutral ($y_\tau = 0$); red: negative ($y_\tau < 0$). Blue vertical line indicates thresholds $\beta_1, \beta_2$ used for selective prediction. 7.2ii Risk-coverage plot for the predictions of the preference model. $\beta_1, \beta_2$ (green lines) are defined to be the thresholds that yield minimum error on the contour of 25% coverage.

Figure 7.3: User 2: 7.3i Predictions of the preference model on sequences from a validation set. Green: sequences labeled with a positive score ($y_\tau > 0$); yellow: neutral ($y_\tau = 0$); red: negative ($y_\tau < 0$). Blue vertical line indicates thresholds $\beta_1, \beta_2$ used for selective prediction. 7.3ii Risk-coverage plot for the predictions of the preference model. $\beta_1, \beta_2$ (green lines) are defined to be the thresholds that yield minimum error on the contour of 25% coverage.
Figure 7.4: User 3: 7.4i Predictions of the preference model on sequences from a validation set. Green: sequences labeled with a positive score ($y_\tau > 0$); yellow: neutral ($y_\tau = 0$); red: negative ($y_\tau < 0$). Blue vertical line indicates thresholds $\beta_1, \beta_2$ used for selective prediction. 7.4ii Risk-coverage plot for the predictions of the preference model. $\beta_1, \beta_2$ (green lines) are defined to be the thresholds that yield minimum error on the contour of 25% coverage.

Figure 7.5: User 4: 7.5i Predictions of the preference model on sequences from a validation set. Green: sequences labeled with a positive score ($y_\tau > 0$); yellow: neutral ($y_\tau = 0$); red: negative ($y_\tau < 0$). Blue vertical line indicates thresholds $\beta_1, \beta_2$ used for selective prediction. 7.5ii Risk-coverage plot for the predictions of the preference model. $\beta_1, \beta_2$ (green lines) are defined to be the thresholds that yield minimum error on the contour of 25% coverage.
Figure 7.6: User Score vs. Beam Width: As we increase the width of the beam, we get a higher score for generated solos using the user preference model. \( x \)-axis - \( b \_k \) combinations of beam width \( b \) and parameter \( k \) used. Shaded area represents the 95% percentile of the confidence interval. Notice the initial point of beam width 1_1 representing the score for improvisations generated by BebopNet without personalization.
Figure 7.7: Percent of common phrases of length $n$-gram length. Our jazz model is in black (BebopNet).

<table>
<thead>
<tr>
<th>Name</th>
<th>Adderley</th>
<th>Gordon</th>
<th>Getz</th>
<th>Parker</th>
<th>Rollins</th>
<th>Stitt</th>
<th>Woods</th>
<th>Ammons</th>
<th>Mean</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adderley</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>4.7</td>
<td>5.7</td>
<td>4.5</td>
<td>5</td>
<td>4.5</td>
<td>5</td>
<td>6.2</td>
</tr>
<tr>
<td>Gordon</td>
<td>3.4</td>
<td>6.4</td>
<td>5.1</td>
<td>4.2</td>
<td>4.6</td>
<td>3.8</td>
<td>3.5</td>
<td>4.2</td>
<td>4.4</td>
<td>4.6</td>
</tr>
<tr>
<td>Getz</td>
<td>3.3</td>
<td>4.6</td>
<td>5.7</td>
<td>4.4</td>
<td>4.2</td>
<td>3.8</td>
<td>3.5</td>
<td>4.2</td>
<td>4.2</td>
<td>4</td>
</tr>
<tr>
<td>Parker</td>
<td>3.7</td>
<td>5</td>
<td>5.1</td>
<td>6</td>
<td>5.1</td>
<td>4.1</td>
<td>3.6</td>
<td>5</td>
<td>4.7</td>
<td>4.3</td>
</tr>
<tr>
<td>Rollins</td>
<td>3.6</td>
<td>4.9</td>
<td>4.6</td>
<td>4.4</td>
<td>4.7</td>
<td>3.8</td>
<td>3.5</td>
<td>4.2</td>
<td>4.2</td>
<td>4.1</td>
</tr>
<tr>
<td>Stitt</td>
<td>4</td>
<td>7</td>
<td>7.2</td>
<td>5.6</td>
<td>5</td>
<td>10.3</td>
<td>4.1</td>
<td>5.6</td>
<td>6.1</td>
<td>4.7</td>
</tr>
<tr>
<td>Woods</td>
<td>4.1</td>
<td>5</td>
<td>5.8</td>
<td>4.8</td>
<td>5.4</td>
<td>4.4</td>
<td>5.4</td>
<td>5.1</td>
<td>5</td>
<td>3.8</td>
</tr>
<tr>
<td>Ammons</td>
<td>3.3</td>
<td>4.8</td>
<td>4.8</td>
<td>3.9</td>
<td>4.3</td>
<td>4</td>
<td>3.6</td>
<td>5.3</td>
<td>4.2</td>
<td>3.9</td>
</tr>
<tr>
<td>Mean</td>
<td>3.7</td>
<td>5.5</td>
<td>5.5</td>
<td>4.7</td>
<td>4.9</td>
<td>4.8</td>
<td>4</td>
<td>4.8</td>
<td>-</td>
<td>4.4</td>
</tr>
<tr>
<td>Ours</td>
<td>2.7</td>
<td>3.9</td>
<td>3.8</td>
<td>3</td>
<td>3.4</td>
<td>2.8</td>
<td>2.8</td>
<td>3.6</td>
<td>3.3</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Table 7.5: Plagiarism among 8 jazz saxophone giants. Element $x_{a,b}$ in the table is the average largest sub-sequence in a solo of artist $a$ (row names) found in any solo of artist $b$ (column names).
7.5 Jazz Dataset (XML files)

2 Autumn Leaves Sonny Stitt Good Life
003 Tenor Madness Coltrane Rollins Prestige
004 Serpents Tooth Take 1 Rollins and Bird
5 Serpents Tooth Rollins Bird Take 2
9 Autumn Leaves Stitt Ammons Boss Tenors
12 Scrapple from the Apple Dexter Gordon Our Man in Paris
15 Just Friends Bird with Strings
38 Summertime Stitt A Jazz Message
40 I've Got Rhythm Stitt Tune up
073 P208-209 GIRL FROM IMPANEMA GETZ
85 My Man Benny Woods
115 Blue Seven Sonny Rollins
126 I Remember You Charlie Parker Verve
134 God Bless The Child Rollins
150 Four Rollins Live
156 Cheese Cake Dexter Gordon Go
162 I Want More Dexter Gordon
163 For Regulars Only Alt Dexter Gordon
164 Airgin Sonny Rollins
173 I Cant Give You Anything But Love Stitt
174 Soon Cannonball
212 Webb City Sonny Stitt Constellation
218 Embraceable You Phil Woods Cool Woods
219 Yesterdays Phil Woods Just Friends
224 Out of Nowhere Stan Getz Roost
228 Jumpin the Blues Charlie Parker J McShann
239 Pennies From Heaven Stan Getz Roost
242 St Thomas Sonny Rollins
255 Ornithology Charlie Parker Roost Live
267 Im Forever Blowing Bubbles Charlie Parker J McShann
285 Cherokee Charlie Parker Trio
288 My Heart Tells Me Charlie Parker
295 Walkin Phil Woods This is How I Feel About Jazz
299 Indian Summer Phil Woods The NY Scene
300 Groovin High Charlie Parker
311 Things Are Getting Better Cannonball Adderley
348 Im A Fool To Want You Dexter Gordon Blue Note
361 Moritat Sonny Rollins Saxophone Collossus
381 Heres That Rainy Day Stan Getz for Lovers
382 Cry Me A River Dexter Gordon Blows Hot and Cool
391 On a Slow Boat To China Stan Getz Soul Eyes
394 Line For Lyons Stan Getz Chet Baker Live in Sweden part1
394 Line For Lyons Stan Getz Chet Baker Live in Sweden part2
395 Ernies Tune Dexter Gordon Blue Note
404 Daddy Plays The Horn Dexter Gordon
408 Everybodys Somebodys Fool Dexter Gordon Blue Note
424 Dont Explain Dexter Gordon A Swingin Affair
428 This Cant Be Love Stan Getz in HiFi
429 There Will Never Be Another You Sonny Sitt Roost
430 Ratio Sonny Stitt In The Beginning
453 Ornithology Sonny Stitt Stitt play Bird
458 As Time Goes By Dexter Gordon Round Midnight
469 Our Love is Here to Stay Phil Woods Jazz For The Carraige Trade
480 April in Paris Charlie Parker with Strings
529 Sunshower Stan Getz Ballads n Bossas The Lost Sessions
534 E Luxo So Stan Getz Jazz Samba
536 Hush a Bye Stan Getz Soul Eyes
551 O Grande Amor Stan Getz Sweet Rain
557 Manha de Carnaval Getz Big Band Bossa Nova
560 Stans Blues Getz Gilberto 2
594 Billies Bounce Charlie Parker Take 5
622 Nows The Time Take 3 Bird Savoy
625 Tanya Dexter Gordon 1 Flight Up
627 Nows The Time Take 4 Bird Savoy
638 So Danco Samba Stan Getz w Gilberto
643 Clear Cut Boogie Rollins Global Warming
644 I remember You Gentle Jug Gene Ammons
659 Star Eyes Dexter Art of the Ballad
670 Vivo Sohando Stan Getz Gilberto
679 Im Just Waiting on A Friend Sonny Rollins Stones Tattoo You
690 WNEW Stan Getz and Bill Evans
712 Laura Charlie Parker With Strings
722 Nobody Else But Me Stan Getz Plays
738 Hanky Panky Dexter Gordon Blue Note
741 Doralice Stan Getz Gilberto
763 Corcovado Cannonball Adderleys Bossa Nova
766 Moonlight In Vermont Stan Getz for Lovers
787 I Know That You Know Rollins Sonny Side Up
793 Salvador Sonny Rollins This is What I Do
797 Blue Room Sonny Rollins Rogers Hart Songbook
819 Walkin Bass Phil Woods plays Henry Mancini
820 On The Sunny Side of the Street Sonny Stitt Sunny Side Up
826 Three Oclock on the Morning Dexter Gordon Go
836 Voce e Eu Stan Getz Gilberto 2
840 Corcovado Stan Getz w Guest Laurindo Almeida
852 Everything Happens to Me Charlie Parker with Strings
861 Exactly Like You Stan Getz Ballads
867 Have Yourself a Merry Christmas Dexter Gordon
871 Counter Clockwise Gene Ammons Boss Tenors
871 Counter Clockwise Stitt Jug Boss Tenors
914 Love Jumped Out Stan Getz Recorded Fall 61 - Tenor Sax
935 Summertime Stan Getz The Definative
953 Moose The Mooch Charlie Parker
982 Anthropology Charlie Parker
988 Dewey Square Charlie Parker
1006 Boston Bernie Dexter Gordon Long Tall Dexter
1019 Meditation Dexter Gordon The Art of the Ballad
1039 Girl from Ipanema Getz Live TV Show
1052 You Talk the Talk Gene Ammons Greatest Hits of 70s
1054 Outra Vez Stan Getz 1984 w guest artist Laurindo Almeida
1059 Of Thee I Sing Stan Getz West Coast Jazz 1955
1065 Blues For Alice Charlie Parker
1070 KC Blues Charlie Parker
1078 On a Slow Boat to China Getz Last Recording
1129 Manha De Carneval Dexter Gordon Gettin Around
1132 Summertime Charlie Parker With Strings
1137 Satin Doll Gene Ammons Organ Combos
1144 Jordu Stan Getz Jazz Masters 8
1152 The Christmas Song Dexter Gordon The Panther
1164 Our Love is Here to Stay Dexter Gordon Blue Note
1167 Red Top Gene Ammon Johnny Coles Savoy
1171 Dr Wu Phil Woods Katy Lied
1182 Flick of a Trick Dexter Gordon Gettin Around
1184 I Want to be Happy Stan Getz w Oscar Peterson
1186 Kateas Dance Gene Ammons Legends of Acid Jazz
1193 But Not For Me Stan Getz Quintessence V1
1194 One Note Samba Stan Getz au Go Go
1197 Samba Triste Stan Getz Rio For Lovers
1202 Fuzzy Gene Ammons Savoy Sessions
1212 As Time Goes By Dexter Gordon Manhattan Symphony
1260 Para Machucar meau Coracao Stan Getz Gilberto
Anything Goes Stan Getz Mulligan Meets in HiFi Getz
As I Live and Bop Stan Getz Complete Studio Sessions
Autumn In NY Dexter Gordon Daddy Plays the Horn
Autumn Leaves Stan Getz Best of the Roost Years
Autumn Leaves Stan Getz & Kenny Barron
Bikini Dexter Gordon 1943-1947
Blow Mr Dexter Gordon 1943-1947
Blowin in the Wind Stan Getz
Blowing Reds Top Gene Ammons 1947-1949
Blue Bossa Dexter Gordon Biting the Apple
Blue Monk Dexter Gordon Live at Montreaux
Blue n Boogie Bird Benedetti d1t1
Blues for Bags Sonny Stitt Only the Blues
Bluing Sonny Rollins Complete Prestige
But Not For Me Gene Ammons Soul Summit
Bye Bye Blackbird Gene Ammons God Bless Jug & Sonny
Bye Bye Blackbird Sonny Stitt Gene Ammons God Bless Jug Sonny
Canadian Sunset Gene Ammons Boss Tenor
Chromatic Aberration Dexter Gordon 43-47
Close Enough For Love Stan Getz The Dolphin
Compulsion Charlie Parker Collectors Item
Compulsion Charlie Parker Sonny Rollins Collectors Item
Concentration Gene Ammons 1947-49
Conception Sonny Rollins Dig
Confirmation Dexter Gordon Daddy Plays the Horn
Cool Cool Daddy Gene Ammons Etta Jones Lonely and Blue
Corcovado Stan Getz Compact Jazz
Crazeology Charlie Parker Take 4 12.17.1947
Crazy Chords Stan Getz 2-fer
Crazy Mary Gene Ammons Free Again
Dancing in the Dark Charlie Parker with Strings
Denial Sonny Rollins Dig
Detour Ahead Stan Getz for Lovers
Dexters Minor Mad Dexter Gordon 1943-1947
Diaper Pin Stan Getz Complete Studio Sessions
Dig Sonny Rollins Jackie McClean Dig - Tenor Sax.
Do What You Do, Do Stan Getz Bossa Nova Years
EAAK Blues Gene Ammons 47-49
Early Autumn Stan Getz 3 Herds 1948
Feijoada Stan Getz Stuttgart 1989
Fools Rush In Stan Getz 1952-1953
For You Sonny Stitt Night Letter
Ginza Samba Stan Getz with Cal Tjader
Going for the Okey Doak Gene Ammons 47-49
Groovin High Charlie Parker Radio 3.23.53 Milt Buckner Trio
Groovy Sambas Cannonball Adderley Bossa Nova
Hairy Sonny Stitt Night Letters
How Deep is the Ocean Stan Getz 1952-1953
I Was Doing Alright Dexter Gordon Doin Alright
Idaho Gene Ammons 47-49
In a Sentimental Mood Sonny Rollins with the MJQ
Indian Summer Stan Getz Quartets
Interlude in Bebop Stan Getz Complete Studio Sessions
It’s Only a Paper Moon Sonny Rollins Complete Prestige Sessions
Its Allright With Me Sonny Rollins Workout
Ive Got You Under My Skin Stan Getz Quartets
Ive Grown Accustomed to her Face Brookmeyer Getz B and Friends - Tenor Sax
Ive Told Evry Little Star Sonny Rollins & the Contemporary Leaders
Joy Spring Stan Getz The Dolphin
Jungle Strut Gene Ammons Brother Jug
Just Friends Charlie Parker Cafe Society 1950
Kong Neptune Dexter Gordon One Flight Up
Lady Bird Dexter Gordon Youtube
Landslide Dexter Gordon Dexter Calling
Leaping Leo Gene Ammons Leo Parker LP 1947-1950 Gene Ammons
Lets Fall in Love Stan Getz Gerry Mulligan Meets in HiFi part1
Lets Fall in Love Stan Getz Gerry Mulligan Meets in HiFi part2
Like Someone In Love Stan Getz The Steamer
Lion Roars Gene Ammons L. Parker 1947-1950
Lullaby of Birdland Stan Getz 1952-1953
Man With a Horn Stan Getz Best of Anita O’day
McDougals Sprout Gene Ammons 1947-1949
Misty Dexter Gordon Montmarte Jazzhus 1965
Moonglow Gene Ammons Up Tight
Moose The Moch Charlie Parker at Storyville 031053
Motens Swing Sonny Stitt Sits in w Oscar Peterson Trio
My Little Suede Shoes Charlie Parker Verve
My Old Flame Sonny Rollins Complete Prestige Sessions
My Romance Gene Ammons Boss Tenor
O Morro Nao Tem Vez Stan Getz Jazz Samba Encore
O Pato Stan Getz
On Rainy Afternoons Stan Getz Children of the World
On a Slow Boat to China Sonny Rollins Prestige Profiles
Our Love is Here to Stay Stan Getz & his Cool Sounds
Out of the Blue Sonny Rollins Dig
Pagan Love Song Gene Ammons Bossa Nova
Pennies From Heaven Stan Getz Complete Studio Session w Jimmy Rainey
Prezervation Stan Getz 2-fer
Rainbow People Dexter Gordon Tower of Power
Red Top Gene Ammons 47-49
Scrapple From the Apple Gerry Mulligan Meets Getz in HiFi
Smile Dexter Gordon Dexter Calling
Soul Shack Sonny Stitt Night Letter
Split Kick Stan Getz Roost Quartets
St Thomas Sonny Rollins You Tube
St Thomas Sonny Stitt Brothers 4
Street Tattoo Stan Getz Cade del Mar
Sugar Coated Gene Ammons 47-49
Tangerine Gene Ammons Jug
Tenderly Dexter Gordon
Tenor Eleven Gene Ammons 1949-1950
That Old Feeling Stan Getz Getz Meets Mulligan in HiFi
The Breeze and I Gene Ammons Up Tight
The Chase Dexter Gordon The Chase Dexter Gordon & Gene Ammons
The Lady in Red Stan Getz Quartets
The Rubaiyat Dexter Gordon Citizen Bop
The Shadow of Your Smile Dexter Gordon A Day in Copenhagen
The Way You Look Tonight Stan Getz Complete Studio Sessions
There is No Greater Love Stitt Gene Ammons Boss Tenors 61
Theres a Small Hotel Stan Getz Quartets
Time On My Hands Stan Getz Plays
Too Close For Comfort Stan Getz Gerry Mulligan Meets in HiFi part1
Too Close For Comfort Stan Getz Gerry Mulligan Meets in HiFi part2
Too Marvelous For Words Stan Getz Quartets
Watermelon Man Dexter Gordon Freddie Hubbard Takin Off - Dexter
Watermelon Man Dexter Gordon Original Hits
Wave Dexter Gordon Quartet
Whats New Dexter Gordon 1963 YT
Whats New Stan Getz Quartets
Why Don’t I Sonny Rollins Blue Note
Windows of the World Stan Getz What the World Needs Now
You Can Depend on Me Dexter Gordon Daddy Plays the Horn
Bibliography


63

Technion - Computer Science Department - M.Sc. Thesis MSC-2021-03 - 2021


כדידלבצעאופטימיזציהשלאלתורייהוגאבעזראות maxlength.אםמקוינ.WriteAllמודריים.אםמקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוינ全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域מקוイン全域麦克
משרר המחשבים, חוקרים ואנשי מוזיקה מתמשכים בשזרז יוזמות לשﻫזות לשперед, מבית להנחת תקציר
משחרר מחשבים, חוקרים ואנשי מוזיקה מעוניינים להשתמש בהם לייצור צורות שונות של אמנויות ובעיקר
ליצירת מוזיקה, וככל הנראה יש לה buổiת دائمitat התהמונות בקרב נגני מוזיקה ממגוון, במגוון
FileType וצורות של ייצוגים מתאימים יותר מממדים אחדים של תכنية verwenden עיבוד שירונות מתאימה.isLoading
כון יום, סרטיים ומוקרנים, ובtrer אחר שיאפשר ליצירת מוזיקהなく התחום. כל יום,
ליצירת האמנים, שונים מה rozwiązותCLR truths של המתכני של כל משטח, וב orderId
במסגרת לאולטרה לאומנות, סרטיים, מוזיקה הרומנטית ומנונים אחרים. למעט הנוגע
וזה נגרב והשדרה של תרבותה לייצר את.o גרם לאחד ואחרים, ישרים רמות בבלוק
骨骼 quicker than.

השיטה המקובלת לייצור מוסיקליים היא של ייצוח עיבודיםיהם ידיעה ובמידה
השיטה לשיטה כימית בבלוק, בלא 하지만, מילוי, תגיות מוזיקליים, או שיטות חדשניות בתשלום
הרב. לפי זה, נושאний ליצוזים בחלונות הדרכים ממוגע ידי שניים, אך לכל ליצור רのように
ספלק מKeySpecיymphonic ת plutosucky על דатурית הלוחanganese של המים ממקהל בבלוק
שלל תחתיים בבלוקמציעות שהזזורים. ממקהל ממקהל, משקף להזזורים Los ספלק מと共に
הערה:купוניות בבלוקמציעות שהזזורים. צוראם, או שמש בבלוקמציעות שהזזורים. בסופו
שלל תחתיים בבלוקמציעות שהזזורים Los זה, משמש בבלוקמציעות שהזזורים. בסופו
שלל תחתיים בבלוקמציעות שהזזורים Los זה, משמש בבלוקמציעות שהזזורים.

בפרק הרדואוט של עיבוד מוחי, מסופר על תרשים של מחושב ממקהל אויב בשפה
מעניין, ממקהל אויב בשפה
לתקנת משקול הפיחים בבזקוס אפקטיביות Zubor העבר משקולות משקולות, פיליפיני
התרמה האלקטטרית של מחושב אויב בברון. זה, ומרכים של תרשים של מחושב אויב בשפה
אנו, ואתו על ידי משקולות פוליפונים, ממקהל אויב בשפה
לתקנת משקול הפיחים בבזקוס אפקטיביות Zubor העבר משקולות משקולות, פיליפיני
התרמה האלקטטרית של מחושב אויב בברון. זה, ומרכים של תרשים של מחושב אויב בשפה
אנו, ואתו על ידי משקולות פוליפונים, ממקהל אויב בשפה
לתקנת משקול הפיחים בבזקוס אפקטיביות Zubor העבר משקולות משקולות, פיליפיני
התרמה האלקטטרית של מחושב אויב בברון. זה, ומרכים של תרשים של מחושב אויב בשפה
אנו, ואתו על ידי משקולות פוליפונים, ממקהל אויב בשפה
לתקנת משקול הפיחים בבזקוס אפקטיביות Zubor העבר משקולות משקולות, פיליפיני
התרמה האלקטטרית של מחושב אויב בברון. זה, ומרכים של תרשים של מחושב אויב בשפה
אנו, ואתו על ידי משקולות פוליפונים, ממקהל אויב בשפה
לתקנת משקול הפיחים בבזקוס אפקטיביות Zubor העבר משקולות משקולות, פיליפיני
התרמה האלקטטרית של מחושב אויב בברון. זה, ומרכים של תרשים של מחושב אויב בשפה
אנו, ואתו על ידי משקולות פוליפונים, ממקהל אויב בשפה
לתקן משקול הפיחים בבזקוס אפקטיביות Zubor העבר משקולות משקולות, פיליפיני
התרמה האלקטטרית של מחושב אויב בברון. זה, ומרכים של תרשים של מחושב אויב בשפה
אנו, ואתו על ידי משקולות פוליפונים, ממקהל אויב בשפה
לתקן משקול הפיחים בבזקוס אפקטיביות Zubor העבר משקולות משקולות, פיליפיני
התרמה האלקטטרית של מחושב אויב בברון. זה, ומרכים של תרשים של מחושב אויב בשפה
אנו, ואתו על ידי משקולות פוליפונים, ממקהל אויב בשפה
לתקן משקול הפיחים בבזקוס אפקטיביות Zubor העבר משקולות משקולות, פיליפיני
התרמה האלקטטרית של מחושב אויב בברון. זה, ומרכים של תרשים של מחושב אויב בשפה
אנו, ואתו על ידי משקולות פוליפונים, ממקהל אויב בשפה
לתקן משקול הפיחים בבזקוס אפקטיביות Zubor העבר משקולות משקולות, פיליפיני
התרמה האלקטטרית של מחושב אויב בברון. זה, ומרכים של תרשים של מחושב אויב בשפה
אנו, ואתו על ידי משקולות פוליפונים, ממקהל אויב בשפה
לתקן משקול הפיחים בבזקוס אפקטיביות Zubor העבר משקולות משקולות, פיליפיני
התרמה האלקטטרית של מחושב אויב בברון.
יצירת אלותוריי' ג'אז באמהצעת רשתות

гибורים עמודים

היבר על מחקר

לשם مليני חקל של הדרישת לקבלת התואר
מוניטור למדעים במדעי המחשב

שונית חביב חכימי

הוג של חברת התיכוניי – מוכן טובנו ליווה
כسأل התשפ"א החיפה נובמבר 2020
יצירת אלגוריתים ל’אור באתרי רשתות נוירונים עמוקים

שונית חביב חכימי