Explicitly Conditioned Melody Generation: A Case Study with Interdependent RNNs

Benjamin Genchel
Center for Music Technology
Georgia Institute of Technology
Atlanta, GA 30332 USA
bgenchel3@gatech.edu

Ashis Pati
Center for Music Technology
Georgia Institute of Technology
Atlanta, GA 30332 USA
ashis.pati@gatech.edu

Alexander Lerch
Center for Music Technology
Georgia Institute of Technology
Atlanta, GA 30332 USA
alexander.lerch@gatech.edu

Abstract

Deep generative models for symbolic music are typically designed to model temporal dependencies in music so as to predict the next musical event given previous events. In many cases, such models are expected to learn abstract concepts such as harmony, meter, and rhythm from raw musical data without any additional information. In this study, we investigate the effects of explicitly conditioning deep generative models with musically relevant information. Specifically, we study the effects of four different conditioning inputs on the performance of a recurrent monophonic melody generation model. Several combinations of these conditioning inputs are used to train different model variants which are then evaluated using three objective evaluation paradigms across two genres of music. The results indicate musically relevant conditioning significantly improves learning and performance, and reveal how this information affects learning of musical features related to pitch and rhythm. An informal subjective evaluation suggests a corresponding improvement in the aesthetic quality of generations.

1 Introduction

With recent advances in deep learning, the generation of symbolic music (i.e., music scores) using deep generative models has become quite popular. Common practice is to model music as a sequence and use some kind of Recurrent Neural Network (RNN) to learn the temporal dependencies using a corpus of symbolic music data.

In its simplest form, this amounts to training a model to predict the next note/event given the previous notes/events. Inputs to these models are typically either discrete tokens or one-hot vectors extracted from raw symbolic music data with limited or no explicit musical context. For example, monophonic melody generation systems typically attempt to learn melody while ignoring any form of explicit harmonic context, which is an important guide in note selection. While polyphonic music generation systems operating on piano roll-like formats include this information implicitly, the data often lacks harmonic classification and generalization — explicit information integral to human composers such as inversion, voicing, and harmonic function are treated as separate, unconnected events whose abstractions the model is expected to learn on its own. Rhythmic considerations like pulse, syncopation, and meter that can play a deciding role in how musical phrases are constructed, are left out. The expectation in these cases is that the deep networks are powerful enough to identify patterns in the data and learn to encode abstract musical concepts such as harmony, meter, and rhythm which is a considerable challenge even for humans.

In instances where researchers have tried to facilitate learning by explicitly conditioning models with additional music-specific information such as chords (Eck and Schmidhuber 2002; Yang, Chou, and Yang 2017) and bar-positions (Hadjeres and Nielsen 2018; Trieu and Keller 2018), these conditionings have been added to the base architecture as a means of improving overall performance, rather than tested as independent variables that potentially affect particular outcomes.

We aim to bridge this gap by explicitly and independently representing various musical components and comparing their effects on the learning and performance of a deep monophonic melody generation system by using them as conditioning inputs. Towards this end, we train and compare the performance of a model consisting of a pair of RNNs which independently model note pitch and note duration sequences conditioned on combinations of musical conditioning inputs. These inputs include (a) inter-conditioning the generation of the pitch and duration networks on each other, (b) conditioning with the chordal harmony of current and next notes, and (c) conditioning with relative bar position. In total, thirteen different configurations are tested, one for each possible combination of conditioning inputs including the case of no conditioning, and evaluated using three different objective evaluation metrics using datasets from two different genres of music: (a) Scottish and Irish folk, and (b) Bebop Jazz.

2 Related Work

2.1 Musical Sequence Modeling Using RNNs

RNNs have been ubiquitously applied to sequence modeling tasks, and symbolic music generation is no exception (Mozer 1994; Franklin 2004; Eck and Schmidhuber 2002; Colombo et al. 2016; Chu, Urtasun, and Fidler 2016). Other symbolic music generative models, such as Variational Auto-Encoders (VAE) and Generative Adversarial Networks (GAN), have

also included RNNs as integral building blocks (Roberts et al. 2018; Yu et al. 2017).

The earliest work on symbolic music generation using neural networks used an architecture designed to take pitch, duration, and chord information as input (Mozer 1994). The model was inspired by psycho-acoustics, towards which hand-designed embeddings were used for pitches, durations and chords. Most modern deep learning architectures have since adopted trainable embedding layers to learn embeddings for the discrete data symbols based on the raw data.

Eck and Schmidhuber were the first to use a Long-Short Term Memory (LSTM) based RNN to generate blues chord progressions with improvised melodies (Eck and Schmidhuber 2002). Their experiments generated polyphonic music, and used fixed-length bins to discretize time.

2.2 Learning Musical Dependencies

Studies in the past have attempted to learn dependencies between musical features by either training and then combining separate networks to learn these features independently or by applying them as conditioning inputs.

Chu et al. encoded prior musical knowledge in a hierarchically structured RNN (Chu, Urtasun, and Fidler 2016) for generating symbolic scores. They first generate melodies and then use those to generate chords.

MIDINet (Yang, Chou, and Yang 2017) includes explicit chord conditioning, assigning a single triad chord per bar. JazzGAN also makes use of explicit chord conditioning, assigning a chord to each half bar in their corpus using 3 distinct conditioning methods corresponding to 3 distinct rhythm representations tested in the authors' experiments (Trieu and Keller 2018). Notably, in this work we use an identical chord representation to that used by JazzGAN (section 3.2).

Colombo et al. proposed an architecture similar to ours, modeling pitch and duration with two separate RNNs (Colombo et al. 2016). They conditioned their pitch network with the previous pitch and note duration and the duration network with the current pitch and duration.

Johnson et al. similarly developed a model that integrates separate networks, each of which trains on the same musical data represented differently (Johnson, Keller, and Weintraut 2017). Each constituent network outputs a distribution over notes/note choices for each timestep; the weighted combination of which is considered as the final output of the model. In addition, the authors condition their model using chordal harmony and metric information.

Conklin and Witten's *Multiple Viewpoint Systems for Music Prediction* derives several sequences of viewpoints, or musical attribute sets, from a sequence of notes in a melody and learns a model for each viewpoint independently. The predictions of different viewpoints are combined later into a single final prediction (Conklin and Witten 1995). Cherla et al. developed a Restricted Boltzmann Machine model based on this approach, predicting pitch events using two viewpoints — note pitch and note duration (Cherla et al. 2013).

Mogren's C-RNN-GAN learns to jointly predict real-valued tuples of frequency, length, intensity and timing, though does not include any chord conditioning (Mogren 2016).

MuseGAN, a system for multi-track music generation, also attempts to learn dependencies between different parts of a song (Dong et al. 2018). The input tracks are represented as piano roll MIDI sequences, and a convolutional GAN based architecture is used to learn dependencies between them in three different configurations.

2.3 Data representation for symbolic music

There exist several choices for representing symbolic music data. For pitches, the two most popular approaches are: (a) One-hot vectors denoting distinct pitches (Colombo et al. 2016), or (b) Embeddings from a learnable embedding layer that maps pitches to lower dimensional vectors (Roberts et al. 2018; Hadjeres, Pachet, and Nielsen 2017). The latter is inspired by word2vec embeddings (Mikolov et al. 2013), which have been used extensively by the natural language processing and machine translation communities.

The most common approach to representing note duration is to split the musical score into fixed-length ticks and encode each tick as an event or set of events (e.g. note held, rest, song end) (Roberts et al. 2018; Hadjeres, Pachet, and Nielsen 2017). While this approach helps maintain a timing-grid and has been used successfully in many studies, it has limitations. For instance, most current models using this approach ignore durations shorter than a sixteenth note. This is because representing a finer time resolution supporting shorter notes or uneven time divisions (e.g., triplets) would increase sequence lengths significantly, which makes learning with RNNs harder.

3 Method

In this study, we aim to analyze the effects of different explicit conditioning inputs on the performance of a recurrent monophonic melody generation system. Specifically, we are interested in how these conditioning inputs impact the prediction of the next note, i.e., predicting the pitch P_t and the duration D_t of the t^{th} note.

3.1 Approach

Note pitch and note duration sequences of monophonic melodies are modelled with two parallel LSTM-RNN networks, one for pitch and one for duration. In the experiments, each of these networks is conditioned with various conditioning inputs, the combinations of which define what will hereafter be refered to as conditioning configurations. Before delving deeper into the model architecture and configurations, the data representation used is described.

3.2 Data Representation

Each monophonic melody M is represented as a sequence of pitch and duration tokens corresponding to itsconstituent notes. Several additional sequences are also constructed from the raw data as described below (see Fig. 1 for an illustration):

(a) **Pitch Sequence,** P: Pitch tokens comprise of 88 notes of a standard piano keyboard and a special token for rests.

(b) **Duration Sequence,** D: Duration tokens are assigned based on a dictionary which maps each of 19 possible note durations to a duration token.

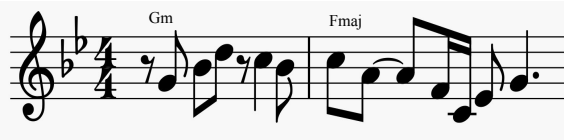

Figure 1: Illustration of data representation scheme. P: pitch token sequence, D: duration token sequence, B: relative bar-position sequence, and C: harmony sequence

(c) **Harmony Sequence,** C: For harmony, the chord type is encoded using a 12-dimensional binary vector with the active pitch classes in the chord. The root pitch is encoded using a separate token.
(d) **Bar-Position Sequence,** B: Tokens indicating the relative position of each note within a bar. Each beat is divided into 24 equal divisions (this ensures that we can adequately represent triplets) resulting in 96 tokens.

3.3 Model Architecture

The base model architecture consists of two parallel LSTM-RNN based networks, denoted as a) *Pitch* Network and b) *Duration* Network, which model the pitch sequence P and the duration sequence D, respectively.

The models are trained to maximize the conditional log-likelihood for the current pitch and duration, given information about the previous notes. For the *Pitch* network, this translates to

$$\max_{\theta_P} \sum_t logprob(P_t|I_{<t}), t \in [0, L-1] \quad (1)$$

where P_t is the pitch of the t^{th} note, $I_{<t}$ is information about all notes that have occurred before the t^{th} note, L is the number of notes, θ_P represents the parameters of the *Pitch* network. Similarly, for the *Duration* network:

$$\max_{\theta_D} \sum_t logprob(D_t|I_{<t}), t \in [0, L-1] \quad (2)$$

where D_t is the duration of the t^{th} note, θ_D represents the paramters of the *Duration* network. All other symbols retain the same meaning as in Equation 1.

Information for the t^{th} note is denoted by I_t and is constructed by concatenating embeddings computed for each input in a given conditioning configuration. A list of the conditioning configurations used is provided in Table 1.

The pitch embeddings p_t, duration embeddings d_t and the bar-position embeddings b_t are computed by processing the elements in P, D and B, respectively, through their respective embedding layers. For chord embeddings c_t and next chord embeddings n_t, the root token of the chords are passed through an embedding layer, while the pitch class vectors are encoded using a set of fully connected layers. The outputs of these layers are then concatenated, and subsequently processed together by a second set of fully connected layers to obtain the final chord embeddings.

Conditioning Configuration	Abbreviation
No conditioning	`No-Cond`
Inter Conditioning	`I`
Chord Conditioning	`C`
Next Chord Conditioning	`N`
Barpos Conditioning	`B`
Chord + Inter	`CI`
Chord + Next Chord	`CN`
Chord + Barpos	`CB`
Inter + Barpos	`IB`
Chord + Next Chord + Inter	`CNI`
Chord + Next Chord + Barpos	`CNB`
Chord + Inter + Barpos	`CIB`
Chord + Next Chord + Inter + Barpos	`CNIB`

Table 1: List of all conditioning configurations. Each conditioning is implemented by concatenating one/more additional embeddings to the input of the networks.

In each conditioning configuration, embeddings for the used conditioning inputs are concatenated with the pitch and duration embeddings to form the information vector I_t. For example, in the `No-Cond` configuration of the *Pitch* network, I_t will only contain the pitch embedding, whereas in the `CI` configuration, I_t will be the concatenation of pitch, current chord, and duration embeddings.

I_t is then encoded by a set of fully connected layers before being passed through the LSTM-RNN. The output of the LSTM-RNN is decoded by a second set of fully connected layers and activations, followed by a Log-Softmax layer. An overview of the model architecture (`CNIB` configuration) is shown in Fig. 2. All other configurations are a subset of this architecture, i.e, they can be derived by selectively removing certain connections prior to the concatenation operation.

4 Experiments

A total of 13 conditioning configurations were used to train models using each of our two datasets of chord annotated lead sheets. All models were implemented and trained using PyTorch. All code including the training and evaluation scripts is available online.[1]

4.1 Datasets

Training and testing of the models is performed using two monophonic melodic datasets.

FolkDB FolkDB comprises of chord-annotated lead sheets in the Scottish and Irish folk tradition. The original dataset[2] is converted from .abc format to MusicXML. Only melodies with $4/4$ time signature and chord annotations were used, resulting in a total of 254 melodies.

BebopDB For the purposes of this study, we created a new dataset of well annotated Bebop Jazz lead sheets in MusicXML format. Each lead sheet contains a monophonic

[1] https://github.com/bgenchel/Explicitly-Conditioning-Melody-Generation

[2] https://github.com/IraKorshunova/folk-rnn/tree/master/data, last accessed: 18th Feb 2019

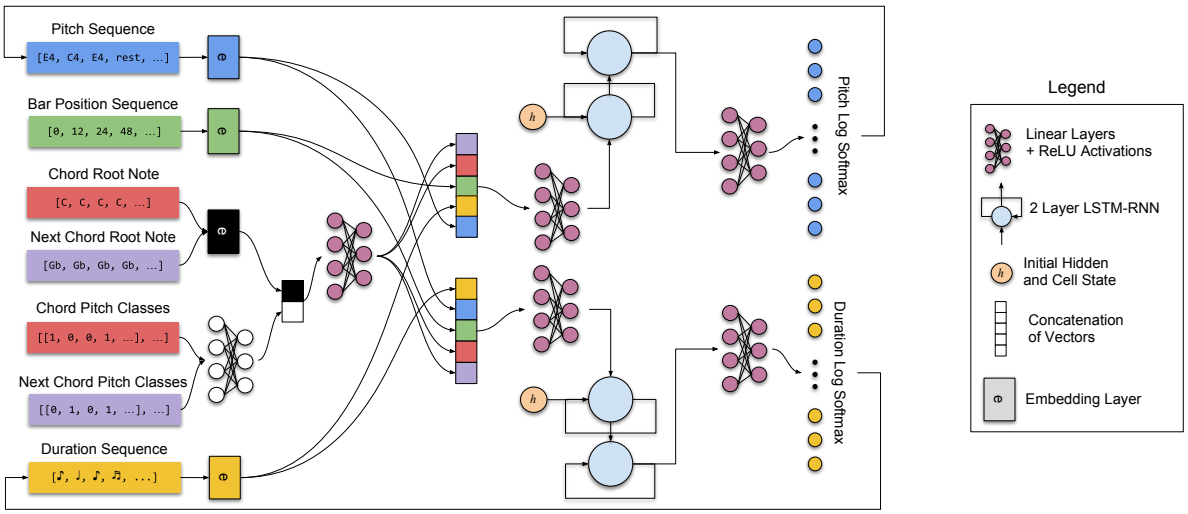

Figure 2: Model architecture overview: This figure displays the `CNIB` configuration of the model, in which all conditioning inputs are all applied simultaneously. Other configurations can be derived from this by selectively removing a subset of connections prior to the embedding concatenation operation. No information from the current timestep is shared between the LSTM-RNNs.

melody with chord annotations, where the chord annotations are placed with specific timing relative to a bar.

At present, the dataset consists of 147 lead sheets from 61 composers. The dataset contains 275 unique chords and 20 unique chord types (e.g., major7, sus4). The longest song in the dataset contains 96 bars, while the shortest contains 12; songs average around 35 bars. The largest number of notes in a bar is 24, and the minimum is 1, with an average of around 6 notes per bar.

Data Pre-processing All melodies are transposed to all root notes by shifting each piece up 5 semi-tones and down 6 semi-tones. This results in 3042 songs for FolkDB and 1617 songs for BebopDB. Transposing the songs, as opposed to shifting each song to the same key, allows the models to better learn songs with key changes.

4.2 Model Specification

Embedding dimensions of 8, 4, 2, and 8 were used for the pitch, duration, chord root, and bar position tokens, respectively. The chord pitch class vector is encoded using linear layers to a dimension of 4. The concatenation of chord root embedding and chord pitch class encoding is further encoded to a dimension of 8.

Linear layers appear in pairs, with the exception of single linear layers feeding into the LSTM-RNNs. The output size of the first layer is the arithmetic mean of its input and the output of the second layer. The layers are separated by a batch-normalization layer (Ioffe and Szegedy 2015) followed by a ReLU activation. Batch-Norm and ReLU layers also follow the single linear layer preceding the LSTM.

Each model uses uni-directional, 2-layer LSTM-RNNs for which both the input size and hidden size are 256-dimensional. The output of the *Pitch* LSTM-RNN is decoded to an 89-dimensional vector, while the *Duration* LSTM-RNN output is decoded to a 19-dimensional vector. Each decoding is fed to a Log-Softmax layer.

4.3 Training Specification

Each model is trained by feeding in sub-sequences of 64 notes ($I_n : I_{n+64}$) and backpropagating the Negative Log-Likelihood Loss (NLL) taken between their outputs and the following sub-sequence of 64 notes from the ground truth data ($I_{n+1} : I_{n+65}$). For training, overlapping windows of length 64 are extracted from each song in the datasets with a hop-size of 1.

Both *Pitch* and *Duration* networks are trained for 30 epochs using a batch-size of 64. We found 30 epochs sufficient for each configuration to converge without overfitting, and thus, used it as an early stopping point. Datasets are divided into training and validation sets using an 80/20% split. We use the AMSGrad variant (Reddi, Kale, and Kumar 2018) of the Adam optimizer (Kingma and Ba 2015) with a learning rate of $1e-3$ Models are regularized by applying dropout to the LSTM layers with 0.2 probability.

5 Evaluation

The performance of the configurations is evaluated using three separate objective metrics: (a) NLL loss achieved on the validation set, (b) performance on the MGEval framework (Yang and Lerch 2018), and (c) BLEU score (Papineni et al. 2002), followed by an informal subjective appraisal of melodies generated by each configuration.

The first metric, NLL, is the most common metric to measure the predictive capability of a generative model and its overall efficiency. The other two, though used to a lesser extent, measure the degree to which the generated melodies comply with the statistics of the training data.

Validation NLL	FolkDB		BebopDB	
Configuration	*Pitch*	*Duration*	*Pitch*	*Duration*
No-Cond	0.373	0.143	0.381	0.136
I	0.303	0.061	0.275	0.065
C	0.317	0.094	0.313	0.085
N	0.319	0.083	0.318	0.075
B	0.277	0.117	0.276	0.112
CI	0.264	0.055	0.241	0.056
CN	0.313	0.075	0.304	0.074
CB	0.240	0.083	0.242	0.073
IB	0.307	0.053	0.274	0.057
CNI	0.248	**0.052**	**0.226**	**0.050**
CNB	**0.212**	0.067	0.239	0.065
CIB	**0.217**	**0.048**	**0.198**	**0.048**
CNIB	**0.206**	**0.047**	**0.190**	**0.045**

Table 2: Best Validation NLL (lower is better) results during training for the *Pitch* and *Duration* networks of different conditioning configurations. Adding different conditioning inputs gradually improves NLL. Adding barposition (B) works better for *Pitch* networks, inter-conditioning (I) works better for *Duration* networks. Bold items denote the top 3.

5.1 NLL Loss

The conditioning configurations are trained to predict the pitch and duration of the current note given information about previous notes. For comparison, we look at the lowest NLL loss obtained on the validation set by each configuration. Table 2 summarizes the results for all different configurations.

Discussion The results suggest that the addition of greater numbers of conditioning inputs improves the predictive performance of both *Pitch* and *Duration* networks on both datasets. This is further supported by a comparison of validation loss curves across training epochs as shown in Fig. 3, where it is clear that more conditioned models converge to lower values faster. This is expected, as the network is provided with more musically relevant context which dictates the flow of the melody. Some observations are:

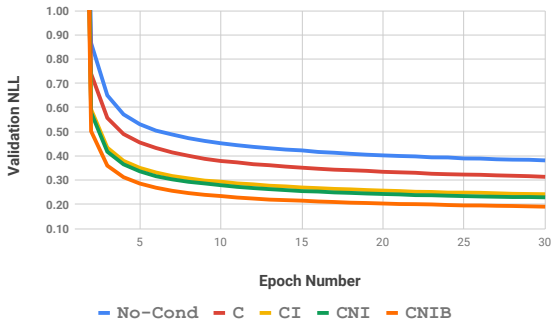

Figure 3: Validation NLL curves for 5 different configurations for the *Pitch* network trained on BebopDB. Adding conditioning inputs improves speed (faster convergence) and performance (lower value) during training.

(a) Adding bar position B significantly improves the performance of *Pitch* networks on this metric for both datasets. This indicates that explicit encoding of this information facilitates better learning relative to implicitly specifying this information by providing only note durations as input. It is interesting to note that bar position does not improve the performance of *Duration* networks when compared to the effects of other inputs such as previous pitch (inter) I, chords C, or next chords N.

(b) Inter-conditioning the *Duration* network improves its performance significantly across both datasets on this metric. This indicates that predicting the duration of the current note relies more heavily on the previous pitch than the current chord, next chord, or bar position.

(c) Providing information regarding the current and next chords without other conditioning inputs (CN case) does not appear to improve performance on this metric, indicating that information regarding the previous pitches/durations and bar positions are possibly more important for learning and prediction.

5.2 MGEval

The MGEval (Music Generation Evaluation) toolbox was designed by Yang and Lerch for objective evaluation of music generation systems (Yang and Lerch 2018). It uses several pitch and duration-based features to measure the degree to which the generated music is able to match the statistics of the training data. The features are modeled as probability distributions. The performance of a generative model is evaluated by computing the distance between these probability distributions across the training data and generated melodies using KL-divergence.

This toolbox is used to compare the performance of each conditioning configuration and to further analyze the impact of each condition (I, C, N, and B). We first generate new melodies using our conditioning configurations for each non-transposed song in both datasets. Melodies are generated by feeding an initial seed of 10 notes to the trained models and then recurrently sampling pitch/duration values from the output Softmax distributions. For models requiring chords, we use the chord progressions from the lead sheets. A set of 11 features, sub-categorized into pitch and duration types, is computed (see Table 3) for both the generated melodies and the original melodies in the datasets. The KL-Divergence between the predicted distribution and inter-set distribution is used as the performance metric.

Table 4 shows the performance of two such features. Since there are a large number of features and conditioning configurations, we use an aggregation technique to summarize the complete set of results.[3] For each feature, the best performing configuration (lowest KL-divergence) is given a score of 1. If there are several configurations with very close KL-divergences (within one standard deviation of the distribution), the score is equally split. Next, the scores are allotted to the individual conditioning inputs based on the conditioning configuration. For instance, for the CNI configuration, the score is split equally amongst C, N and I. The scores for each

[3] https://bgenchel.github.io/ecmg/results

Feature Type	Feature Name
Pitch-based	Pitch Count (PC), Pitch Count/Bar (PC/bar), Pitch Class Histogram (PCH), Pitch Class Transition Matrix (PCTM), Pitch Range (PR), average Pitch Interval (PI)
Duration-based	average Inter-Onset Interval (IOI), Note Length Histogram (NLH), Note Length Transition Matrix (NLTM), Note Count (NC), Note Count/Bar (NC/Bar)

Table 3: List of features used from the MGEval Toolbox. More information about these features and their computation process is given in the original paper (Yang and Lerch 2018).

KL-Divergence	FolkDB		BebopDB	
Configuration	**PR**	**NLH**	**PR**	**NLH**
No-Cond	**0.019**	**1.652**	**0.027**	**2.154**
I	**0.035**	2.297	**0.029**	4.750
C	0.046	2.212	0.047	**2.452**
N	0.050	2.696	0.048	3.717
B	0.052	2.799	0.060	**2.731**
CI	0.094	2.671	0.046	5.347
CN	0.059	3.063	0.061	3.588
CB	0.072	2.273	**0.023**	**2.409**
IB	0.073	2.337	**0.022**	4.471
CNI	0.090	3.368	0.038	5.584
CNB	0.057	2.792	0.050	**2.914**
CIB	**0.033**	**1.439**	**0.030**	6.101
CNIB	0.056	**1.883**	0.041	5.965

Table 4: Predicted-set to inter-set KL-divergence (lower is better) for two MGEval features. PR: Pitch Range, NLH: Note Length Histogram. Bold items are best performers for that feature (within one standard deviation of the top performer).

conditioning input are then added across features within each feature sub-category and normalized to a sum of one. The final normalized scores can be interpreted as the probability that a particular conditioning performs better than others. These normalized distributions are presented in Fig. 4.

Discussions On an average, conditioning inputs improve the performance on the MGEval features. However, different conditioning inputs affect the different feature sub-categories differently. Inter, chord and bar-position clearly perform better for pitch-based features. This is in line with our observations for the NLL loss. There are, however, no clear winners for the duration based features. For FolkDB, next-chord performs the best, while for BebopDB, chord and bar position perform better. Interestingly, for FolkDB, the no-cond case scores the highest for duration-based features. This might be due to the simple rhythmic patterns in that genre.

While these results are interesting, there are also a few concerns. Most prominently, it is assumed that there is an equal contribution from all inputs towards the performance of configuration. To further verify, all scores were normalized for all metrics to have zero mean and unit varience, and

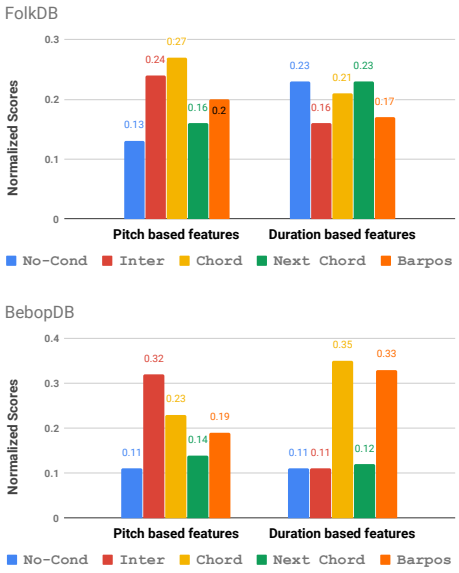

Figure 4: Aggregated score distribution (higher is better) across MGEval sub-categories for each conditioning input

t-tests were conducted for each condition for both pitch-based and rhythm-based features. These tests showed that the results are statistically significant for inter, chord, and next chord conditioning in BebopDB and for chord and next chord conditioning in FolkDB. It is also interesting to note that Table 4 does not display the same trend as Table 2, in which improvement was clearly correlated with increased conditioning. Additionally, the relatively weak performance of inter-conditioning for duration-based features runs counter to the observation of the NLL losses. This indicates that while model training improves with different conditioning inputs, the musical properties of the generated melodies with respect to the training data do not necessarily improve.

5.3 BLEU Score

The BLEU score, a measure of similarity between two corpuses of text, was originally designed to give an objective evaluation for machine translation tasks (Papineni et al. 2002). It is calculated as the geometric mean of the counts of matching n-grams between the generated and target corpuses. A perfect score of 1 is attained only when the two corpuses are exactly the same, which is almost impossible even for humans to achieve on a generative task. Even though the utility of this metric for generative tasks is debatable, BLEU score can still be considered a useful objective metric in the absence of better objective metrics to quantify creativity and musicality. Therefore, it has been used to evaluate music generation systems (Yu et al. 2017).

We compute corpus-level BLEU scores (shown in Table 5) using 1, 2, 3 and 4-gram sequences for the generated melodies for each configuration. The melody generation process is the same as that followed for the MGEval case.

BLEU Score	FolkDB		BebopDB	
Configuration	*Pitch*	*Duration*	*Pitch*	*Duration*
No-Cond	0.267	**0.874**	0.098	**0.875**
I	0.269	0.717	0.101	0.568
C	0.297	0.782	0.133	**0.688**
N	0.311	**0.796**	0.112	0.663
B	0.242	0.763	0.112	**0.696**
CI	0.278	0.685	0.130	0.569
CN	**0.326**	0.779	**0.153**	0.666
CB	0.289	0.784	0.121	0.654
IB	0.253	0.689	0.129	0.607
CNI	**0.326**	0.684	**0.157**	0.572
CNB	**0.335**	**0.794**	**0.146**	0.638
CIB	0.283	0.694	0.132	0.593
CNIB	**0.317**	0.688	0.128	0.566

Table 5: Corpus Level BLEU Scores (higher is better) for *Pitch* and *Duration* networks for different conditioning configurations and datasets. **Bold** items denote the top 3.

Discussion Between the *Pitch* and the *Duration* networks, the scores are more intuitively understandable for the former case. For *Pitch* networks, the highest scores are achieved by configurations which include chords, either current or next. This indicates that adding chords as conditioning inputs helps the network to better reproduce pitch n-grams.

For the *Duration* networks, No-Cond performs the best. This is quite surprising since this configuration only relies on the duration of previous notes. The second and third highest scores come from chord and bar position conditioning. The FolkDB models work better with next chord, while the BebopDB models perform better with the current chord. Interestingly, once again, the lowest scores for duration seem to correspond with inter-conditioning.

The results, especially those for *Duration* networks, run counter to the NLL loss results reported above. However, this tallies with observations from the MGEVal framework and warrants further investigation.

5.4 Subjective Appraisal of Sample Melodies

In order to get a sense of how well the objective measures correspond to our aesthetic experience, we generate a few melodies following the harmony and seed selected randomly from a few lead sheets. While there were some similarities between generations of FolkDB and BebopDB models based on the conditioning configurations, due to musical differences between the genres, some were harder to judge than others.

Models which lack any kind of chord conditioning were in general able to stay in key for both genres for some time. As Folk songs tend to stay within the same key throughout, chord conditioning did not seem necessary for consonance. For Bebop generations however, configurations that lacked chord conditioning, eventually fell out of key and were unable to find their way back.

For Folk generations, the addition of inter-conditioning seemed to induce specific bad habits, such as repeatedly returning to a particular note, or making repeated large jumps to note B4. These were mitigated in combination with chord conditioning, which in general led to higher quality generations. For Bebop generations, we saw no such effect for inter-conditioning in isolation, however, we did observe this synergy when combined with chord conditioning.

Barpos conditioning appeared to enhance rhythmic variety in Bebop generations and increase the duration for which generations stayed in key. The same was not observed for Folk generations. Chord conditioning produced the most noticeable effect for both genres, significantly increasing harmonicity for Bebop generations and improving the musicality of Folk generations. Chord-conditioned Folk generations seemed to include a greater number of intervals such as major and minor thirds and well-placed fourths, which we felt added more emotion and musicality. Without chord conditioning, Folk generations tended to over-use consecutive whole tone (major 2nd) intervals which sounded monotonous. In addition, chord conditioning seemed to have a strong positive impact on rhythmic phrasing and variety for both genres.

Chord conditioning seemed to have a much more positive impact on Bebop generations than next chord conditioning, which sometimes appeared to cause the model to step out of key, perhaps in anticipation of the next chord. For Folk generations, chord and next chord conditioning appeared to similarly improve performance.

Despite the observed effects listed above, we still found that CNIB and CIB produced the most aesthetically pleasing melodies over all, indicating positive synergy between conditionings even when they appear detrimental in isolation. CI, CNI, and CNB were close seconds in quality. These observations are consistent with results from the objective metrics, indicating the importance of harmony on both pitch and rhythmicity, as well as its synergy with inter-conditioning and to a lesser extent bar position. Selected score samples are displayed in Figure 5. We provide audio for these generated melodies for the reader to make their own judgments[4].

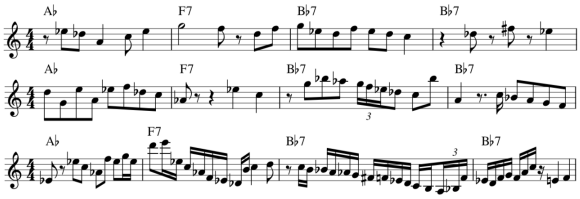

Figure 5: Generations over the first 4 bars of "Donna Lee" by Charlie Parker for (from top to bottom) I, CI, and CIB.

6 Conclusion

We present a case study designed to evaluate and analyze the effects of explicit musical conditioning on monophonic melody generation. We describe thirteen different conditioning configurations of four musical conditioning inputs (inter-conditioning between pitch and duration sequences, current chord, next chord, and bar position), each of which is used to train an LSTM-RNN based model on two datasets of chord-labeled melodies. We evaluate the performance of models

[4] https://bgenchel.github.io/ecmg/results

in each conditioning configuration with three objective measures, providing insight into how these conditioning inputs effect overall learning rate and accuracy and how they facilitate the model in learning musical features from the data. In addition, we also provide a subjective appraisal of the aesthetic qualitty of the melodies generated by models trained using different conditioning configurations.

The results of this study suggest that harmonic conditioning is important not just for pitch prediction, but for duration prediction as well. More generally, it provides some insight into the relative usefulness of each of the conditioning inputs for learning pitch and rhythm-based features. We also discover that while some features may appear ineffective or even detrimental when applied individually, or in certain combinations, they may still be useful when applied in other combinations. In addition, we show that the effectiveness of musical conditioning is also dependent on the type of data used; though there were commonalities in how conditioning effected generations for both Folk-trained and Bebop-trained models, there were significant differences as well.

In the future, we aim to conduct a subjective listening test to formally determine if the results of our objective metrics line up with the aesthetic perception, as well as test a greater number of musical conditioning factors, some of which relate to long term structure.

References

Cherla, S.; Weyde, T.; Garcez, A.; and Pearce, M. 2013. A distributed model for multiple-viewpoint melodic prediction. In *Proc. of International Society of Music Information Retrieval Conference (ISMIR)*, 15–20.

Chu, H.; Urtasun, R.; and Fidler, S. 2016. Song from PI: A musically plausible network for pop music generation. In *International Conference on Learning Representations (ICLR), Workshop Track*.

Colombo, F.; Muscinelli, S.; Seeholzer, A.; Brea, J.; and Gerstner, W. 2016. Algorithmic composition of melodies with deep recurrent neural networks. In *Proc. of the 1st Conference on Computer Simulation of Musical Creativity (CSMC)*.

Conklin, D., and Witten, I. H. 1995. Multiple viewpoint systems for music prediction. *Journal of New Music Research* 24(1):51–73.

Dong, H.-W.; Hsiao, W.-Y.; Yang, L.-C.; and Yang, Y.-H. 2018. MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*.

Eck, D., and Schmidhuber, J. 2002. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In *Proc. of the 12th IEEE Workshop on Neural Networks for Signal Processing*, 747–756.

Franklin, J. A. 2004. Computational models for learning pitch and duration using lstm recurrent neural networks. In *Proc. of the 8th International Conference on Music Perception and Cognition (ICMPC)*.

Hadjeres, G., and Nielsen, F. 2018. Anticipation-RNN: Enforcing unary constraints in sequence generation, with application to interactive music generation. *Neural Computing and Applications* 1–11.

Hadjeres, G.; Pachet, F.; and Nielsen, F. 2017. DeepBach: A steerable model for Bach chorales generation. In *Proc. of the 34th International Conference on Machine Learning (ICML)*, 1362–1371.

Ioffe, S., and Szegedy, C. 2015. Batch Normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of the 32nd International Conference on Machine Learning (ICML)*, 448–456.

Johnson, D. D.; Keller, R. M.; and Weintraut, N. 2017. Learning to create jazz melodies using a product of experts. In *Proc. of the 8th International Conference on Computational Creativity (ICCC)*, 151–158.

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

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 3111–3119.

Mogren, O. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. In *Proc. of Constructive Machine Learning Workshop, NeurIPS*.

Mozer, M. C. 1994. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science* 6(2-3):247–280.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*, 311–318. Association for Computational Linguistics.

Reddi, S. J.; Kale, S.; and Kumar, S. 2018. On the convergence of adam and beyond. In *International Conference on Learning Representations (ICLR)*.

Roberts, A.; Engel, J.; Raffel, C.; Hawthorne, C.; and Eck, D. 2018. A hierarchical latent vector model for learning long-term structure in music. In *Proc. of the 35th International Conference on Machine Learning (ICML)*, 4364–4373.

Trieu, N., and Keller, R. M. 2018. JazzGAN : Improvising with generative adversarial networks. In *Proc. of the 6th International Workshop on Musical Metacreation (MUME)*.

Yang, L.-C., and Lerch, A. 2018. On the evaluation of generative models in music. *Neural Computing and Applications*.

Yang, L.; Chou, S.; and Yang, Y. 2017. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 324–331.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. SeqGAN: Sequence generative adversarial aets with policy gradient. In *Proc. of the 31st AAAI Conference on Artificial Intelligence*, 2852–2858.