# COUPLED RECURRENT MODELS FOR POLYPHONIC MUSIC COMPOSITION

**John Thickstun**[1] **Zaid Harchaoui**[2] **Dean P. Foster**[3] **Sham M. Kakade**[1,2]
[1] Allen School of Computer Science and Engineering, University of Washington, Seattle, WA, USA
[2] Department of Statistics, University of Washington, Seattle, WA, USA
[3] Amazon, NY, USA
{thickstn,sham}@cs.washington.edu, zaid@uw.edu, foster@amazon.com

## ABSTRACT

This paper introduces a novel recurrent model for music composition that is tailored to the structure of polyphonic music. We propose an efficient new conditional probabilistic factorization of musical scores, viewing a score as a collection of concurrent, coupled sequences: i.e. voices. To model the conditional distributions, we borrow ideas from both convolutional and recurrent neural models; we argue that these ideas are natural for capturing music's pitch invariances, temporal structure, and polyphony. We train models for single-voice and multi-voice composition on 2,300 scores from the KernScores dataset.

## 1. INTRODUCTION

In this work we will think of a musical score as a sample from an unknown probability distribution. Our aim is to learn an approximation of this distribution, and to compose new scores by sampling from this approximation. For a broad survey of approaches to automatic music composition, see [9]; for a more targeted survey of classical probabilistic approaches, see [3]. We note the success of parameterized, probabilistic generative models in domains where problem structure can be exploited by models: convolutions in image generation, or autoregressive models in language modeling. This work examines autoregressive models of scores (Section 3): how to evaluate these models, how to build the structure of music into parameterized models, and the effectiveness of these modeling choices.

We study the impact of structural modeling assumptions via a cross-entropy measure (Section 4). It is reasonable to question whether cross-entropy is a good surrogate measure for the subjective quality of sampled compositions. In theory, a sufficiently low cross-entropy indicates a good approximation of the target distribution and therefore must correspond to high-quality samples. In practice, we observe of other generative modeling tasks that learned models do achieve sufficiently low cross-entropy to produce qualitatively good samples [4, 19, 29]. Studying the cross-entropy allows us to explore many models with various structural assumptions (Section 5). Finally, we provide a qualitative evaluation of samples from our best model to demonstrate that these models have sufficiently small cross-entropy for samples to exhibit a degree of subjective quality (Section 6). Supplementary material including appendices, compositional samples, and code for the experiments is available online. [1]



## 2. RELATED WORKS

In this work, we consider both single-voice models and multi-voice, polyphonic models. Early probabilistic models of music focused on single-voice, monophonic melodies. The first application of neural networks to melody composition was proposed by [28]. This work prompted followup [18] using an alternative data representation inspired by pitch geometry ideas [25]; the relative pitch and note-embedding schemes considered in the present work can be seen as a data-driven approach to capturing some of these geometric concepts. For recent work on monophonic composition, see [12, 23, 26].

Work on polyphonic music composition is considerably more recent. Early precurors include [15], which considers two-voice composition, and [5], which proposes an expert system to harmonize 4-voice Bach chorales. The harmonization task became popular, along with the Bach chorales dataset [1]. Multiple voice polyphony is directly addressed in [16], albeit using a simplified preprocessed encoding of scores that throws away duration information.

Maybe the first work with a fair claim to consider polyphonic music in full generality is [2]. This paper proposes a coarse discrete temporal factorization of musical scores (for a discussion of this raster factorization and others, see Section 3) and examines the cross-entropy of a variety of neural models on several music datasets (including the Bach chorales). Many subsequent works on polyphonic models use the dataset, encoding, and quantitative metrics introduced in [2], notably [30] and [13]. We also note recent, impressive work on the closely related problem of modeling expressive musical performances [11, 20].

Many recent works focus exclusively on the Bach

[1] http://homes.cs.washington.edu/~thickstn/ismir2019composition/

chorales dataset [7, 10, 17]. The works [7, 17] evaluate their models using qualitative large-scale user studies. The system proposed in [7] optimizes a pseudo-likelihood, so its quantitative losses cannot be directly compared to generative cross-entropies. The generative model proposed in [17] could in principle report cross entropies, but this work also focuses on a qualitative study. Quantitative cross-entropy metrics on the chorales are analyzed in [10]. Both [7] and [10] propose non-sequential Gibbs-sampling schemes for generation, in contrast to the ancestral samplers used in [17] and in the present work.

## 3. FACTORING THE DISTRIBUTION OVER SCORES

Polyphonic scores consist of notes and other features of variable length that overlap each other in quasi-continuous time. Scores contain a vast heterogenous collection of information, much of which we will not attempt to model: time signatures, tempi, dynamics, etc. We will therefore give a working definition of a score that captures the pitch, rhythmic, and voicing information we plan to model. We define a score of length $T$ beats as a continuous-time, matrix-valued sequence $\mathbf{x}$, where $\mathbf{x}_t \in \{0,1\}^{V\times 2P}$ for each time $t \in [0,T]$. Specifically, for each voice $v \in \{1,\dots,V\}$ and each pitch $p \in \{1,\dots,P\}$ we set

$$\mathbf{x}_{t,v,p} = 1 \quad \text{iff pitch } p \text{ is on at time } t \text{ in voice } v, \quad (1)$$

$$\mathbf{x}_{t,v,P+p} = 1 \quad \text{iff pitch } p \text{ begins at time } t \text{ in voice } v. \quad (2)$$

Both "note" bits (1) and "onset" bits (2) are required to represent a score, expressing the distinction between a sequence of repeated notes of the same pitch and a single sustained note; see Appendix C for further discussion.

Let $q$ denote the (unknown) probability distribution over scores $\mathbf{x}$. Score are high dimensional objects, of which we have limited samples (2,300 – see Section 4). Rather than directly model $q$, we will serialize $\mathbf{x}$, factor $q$ according to this serialization, and model the resulting conditional distributions $q(\cdot|\text{history})$. There are many possible ways to factor $q$; in the remainder of this section we review the popular raster factorization, and propose a new sequential factorization based on voices.

**Raster score factorization.** Many previous works factor a score via rasterization. If we sample a score $\mathbf{x}$ at constant intervals $\Delta$ and impose an order on parts and notes, we can factor the distribution $q$ over scores as $q(\mathbf{x}) =$

$$\prod_{k=1}^{T/\Delta}\prod_{v=1}^{V}\prod_{p=1}^{2P} q(\mathbf{x}_{k\Delta,v,p}|\mathbf{x}_{1:k\Delta}, \mathbf{x}_{k\Delta,1:v}, \mathbf{x}_{k\Delta,v,1:p}). \quad (3)$$

Throughout this work, a slice $1\!:\!i$ is inclusive of the first index 1 but does not include the final index $i$.

This factorization generates music in sequential $\Delta$-slices of time. Some prior works directly model the (high-dimensional) distribution $\mathbf{x}_{k\Delta}$; this approach was pioneered by [2], using NADE to model the conditional distributions $q(\mathbf{x}_{k\Delta}|\mathbf{x}_{1:k\Delta})$. Others impose further order on notes (and voicings, if they choose to model them) and factor the distribution into binary conditionals as in (3). Notes are typically ordered based on pitch, either low-to-high [7] or high-to-low [17].

**Sequential voice factorization.** Putting full scores aside for now, consider factoring a single voice $v$, i.e. a slice $\mathbf{x}_{1:T,v,1:2P}$ of a score. By definition, a Kern-Scores voice is homophonic in the sense that its rhythms proceed in lock-step: a voice consists of a sequence of notes, chords, or rests, and no notes are sustained across a change point. [2] Instead of generating raster time slices, suppose we run-length encode the durations between change points in $v$. We denote these change points by $c_0^v,\dots,c_{L_v}^v$ where $L_v$ is the number of change points in voice $v$. Let $D$ be the number of unique distance between change points, and define a run-length encoded voice $\mathbf{r} \in \left(\{0,1\}^D \oplus \{0,1\}^N\right)^{L_v}$. At each index $k \in \{1,\dots,L_v\}$, $\mathbf{r}_k = (\mathbf{r}_{k,0}, \mathbf{r}_{k,1})$ with $\mathbf{r}_{k,0} \in \{0,1\}^D$ and $\mathbf{r}_{k,1} \in \{0,1\}^N$ such that

$$\mathbf{r}_{k,0} = \mathbf{1}_{d_k} \quad \text{where } d_k = \frac{c_{k+1}^v - c_k^v}{\Delta} \in \mathbb{N},$$

$$\mathbf{r}_{k,1,p} = 1 \quad \text{iff pitch } p \text{ begins at time } c_k^v \text{ in voice } v.$$

The durations $d_k$ correspond to note-values (quarter-note, eighth-note, dotted-half, etc.). We proceed to factor the voice sequentially as $p(\mathbf{r}) =$

$$\prod_{k=1}^{L_v} q(\mathbf{r}_{k,0}|\mathbf{r}_{1:k}) \prod_{p=1}^{P} q(\mathbf{r}_{k,1,p}|\mathbf{r}_{1:k}, \mathbf{r}_{k,0}, \mathbf{r}_{k,1,1:p}). \quad (4)$$

**Sequential score factorization.** We now consider a sequential factorization that interlaces predictions in the score's constituent voices. The idea is to predict voices sequentially as we did in the previous section, but we must now choose the order across voices in which we make predictions. The rule we choose is to make a prediction in the voice that has advanced least far in time, breaking ties by the arbitrary numerical order assigned to voices (ties happen quite frequently: for example, at the beginning of a score when all parts have advanced 0 beats). This ensures that all voices are generated in near lock-step; generation in any particular voice never advances more than one note-value ahead of any other voice.

Mathematically, we can describe this factorization as follows. First, we impose a total order on change points $c_k^v$ across voices by the rule $c_k^v < c_{k'}^u$ for all $v,u$ if $k<k'$ and $c_k^v < c_k^u$ if $v<u$. Define $L \equiv \sum_{v=1}^{V} L_v$. For index $i \in \{1,\dots,L\}$ let $\alpha_i$ and $\beta_i$ denote the index and voice of the corresponding change point according the the total ordering on change points. We define a sequentially encoded score $\mathbf{s} \in (\{0,1\}^D \oplus \{0,1\}^N)^L$ by

$$\mathbf{s}_{k,0} = \mathbf{1}_{d_k} \quad \text{where } d_k = \frac{c_{\alpha_k+1}^{\beta_k} - c_{\alpha_k}^{\beta_k}}{\Delta} \in \mathbb{N},$$

$$\mathbf{s}_{k,1,p} = 1 \quad \text{iff pitch } p \text{ begins in voice } \beta_k \text{ at time } c_{\alpha_k}^{\beta_k}.$$

And we factor the distribution sequentially by $q(\mathbf{s}) =$

$$\prod_{k=1}^{L} q(\mathbf{s}_{k,0}|\mathbf{s}_{1:k}) \prod_{p=1}^{P} q(\mathbf{s}_{k,1,p}|\mathbf{s}_{1:k}, \mathbf{s}_{k,0}, \mathbf{s}_{k,1,1:p}). \quad (5)$$

[2] For polyphonic instruments like the piano, we must adopt a more refined definition of a voice than "notes assigned to a particular instrument;" see Appendix B for details.

## 4. DATASET AND EVALUATION

**Dataset**. The models presented in this paper are trained on KernScores data [24], a collection of early modern, classical, and romantic era digital scores assembled by musicologists and researchers associated with Stanford's CCARH. [3] The dataset consists of over 2,300 scores containing approximately 2.8 million note labels. Tables 1 and 2 give a sense of the contents of the dataset.

We contrast this dataset's Humdrum encoding with the MIDI encoded datasets used by most works discussed in this paper. [4] MIDI was designed as a protocol for communicating digital performances, rather than digital scores. This is exemplified by the MAPS [6] and MAESTRO [8] datasets, which consist of symbolic MIDI data aligned to expressive performances. While this data is symbolic, it cannot be interpreted as scores because it is unaligned to a grid of beats and does not encode note-values (quarter-note, eighth-note, etc). Some MIDI datasets are aligned to a grid of beats, for example MusicNet [27]. But heuristics are still necessary to interpret this data as visual scores. For example, many MIDI files encode "staccatto" articulations by shortening the length of notes, thwarting simple rules that identify note-values based on length.

**Evaluation**. Let $\hat{q}$ be an estimate of the unknown probability distribution over scores $q$. We want to measure the quality of $\hat{q}$ by its cross-entropy to $q$. Because the entropy of a score grows with its length $T$, we will consider a cross-entropy rate. By convention, we measure time in units of beats, so the cross-entropy rate has units of bits per beat.

Defining cross-entropy for a continuous-time process generally requires some care. But for music, defining the cross-entropy on an appropriate discretization will suffice. Musical notes begin and end at rational fractions of the beat, and therefore we can find a common denominator $d$ of all change points in the support of the distribution $q$ (for our dataset $d = 48$). For a score of length $T$ beats, we partition the interval $[0, T]$ into constant subintervals of length $\Delta \equiv 1/d$ and define a rate-adjusted, discretized cross-entropy

$$H_{\mathcal{P}}(q||\hat{q}) \equiv \mathop{\mathbb{E}}_{\mathbf{x}\sim q}\left[-\frac{1}{T\Delta}\log \hat{q}(\mathbf{x}_0, \mathbf{x}_\Delta, \mathbf{x}_{2\Delta}, \ldots, \mathbf{x}_T)\right]. \tag{6}$$

Proposition 1 in Appendix F shows that we can think of $\Delta$ as the resolution of the score process, in the sense that any further refinement of the discretization $d$ yields no further contributions to the cross entropy.

Definition 6 is independent of any choice about how we factor $q$: it is a cross entropy measure of the joint distribution over a full score. As we discussed in Section 3, there are many ways to factor a generative model of scores. These choices lend themselves to different natural cross-entropies, each with their own units. By measuring in units of bits per beat at the process resolution $\Delta$ as defined by Definition 6, we can compare results under different factorizations.

[3] http://kern.ccarh.org/

[4] A notable exception is [16], which uses data derived from the KernScores collection considered here.

**Computational cost**. Raster models are expensive to train and evaluate on rhythmically diverse music. A raster model must be discretized at the process resolution $\Delta$ to generate a score with precise rhythmic detail. The process resolution $\Delta$ of a corpus containing both triplets and sixty-fourth notes is $\Delta = 3 \times 16 = 48$ positions per beat. Corpora with quintuplet patterns require a further factor of 5, resulting in $\Delta = 240$. To generate a score from a raster factorization requires $\Delta$ predictions per beat; to ease the computational burden of prediction, when the raster approach is taken scores are typically discretizing at either 1 or 2 positions per beat [2]. Unfortunately, this discretization well above the process resolution leads to dramatic rhythmically simplification of scores (see Appendix C).

In contrast, a sequential factorization such as (4) or (5) requires predictions proportional to the average number of notes per beat, while maintaining the rhythmic detail of a score. The KernScores single-voice corpus averages $\approx 1.25$ notes per beat, requiring $1.25$ predictions per beat for sequential factorization versus $\Delta$ predictions per beat for raster factorization. The KernScores multi-voice corpus averages $\approx 5$ notes per beat, requiring $5$ predictions per beat for sequential factorization, an order of magnitude less than the $\Delta \approx 50$ predictions per beat required for raster prediction.

## 5. MODELS AND WEIGHT-SHARING

Modeling voices allows us to think of the polyphonic composition problem as a collection of correlated single-voice composition problems. Learning the marginal distribution over a single voice $v$ is similar in spirit to classical monophonic tasks. Learning the distribution over KernScores voices generalizes this classical task to allow for chords: formally, a monophonic sequence would require the vector $\mathbf{r}_{k,1} \in \{0, 1\}^N$ described in Section 3 to be one-hot, whereas our our dataset includes voices where this vector is multi-hot, expressing intervals and chords (e.g. chords in the left hand of a piano, or double-stops for a violin).

We will explore two modeling tasks. First we consider a single-voice prediction task: learn the marginal distribution over a voice $v$, estimating the conditionals that appear in the factorization (4). Results on this task are summarized in Table 3. Second we consider a multi-voice prediction task: learn the joint distribution over scores, estimating the conditionals that appear in the factorization (5). Results on this task are summarized in Table 4.

### 5.1 Representation

Like our choice of factorization, we are faced with many options for encoding the history of a score for prediction. Some of the same computational and modeling considerations apply to both the choice of a factorization and the choice of a history encoding, but these are not inherently connected decisions. For the single-voice task, we use the encoding $\mathbf{r}$ introduced to define the sequential voice factorization in Section 3.

For the polyphonic task, we also encode history using a run-length encoding. Let $c_1, \ldots, c_K$ denote change points in the full score $\mathbf{x}$, let $d_j^v \equiv (c_{j+1}^v - c_j^v)/\Delta \in \mathbb{N}$, and define

| Bach | Beethoven | Chopin | Scarlatti | Early | Joplin | Mozart | Hummel | Haydn |
|---|---|---|---|---|---|---|---|---|
| 191,374 | 476,989 | 57,096 | 58,222 | 1,325,660 | 43,707 | 269,513 | 3,389 | 392,998 |

**Table 1**. Notes in the KernScores dataset, partitioned by composer. The "Early" collection consists of Renaissance vocal music; a plurality of the Early music is composed by Josquin.

| Vocal | String Quartet | Piano |
|---|---|---|
| 1,412,552 | 820,152 | 586,244 |

**Table 2**. Notes in the KernScores dataset, partitioned by ensemble type.

a sequence $\mathbf{e} \in (\{0,1\}^{D+1} \oplus \{0,1\}^{P})^{K \times V}$ where

$\mathbf{e}_{k,v,0,0:D} = \mathbf{1}_{d_j^p}$ iff $c_k = c_j^v$ for some $c_j^v$ in voice $v$,
$\mathbf{e}_{k,v,0,D} = 1$ iff $c_k$ is not a change point in voice $v$,
$\mathbf{e}_{k,v,1,p} = 1$ iff pitch $p$ begins in voice $v$ at time $c_k$.

This is not the fully serialized encoding **s** used to define a score factorization (for discussion of a fully sequential representation, see [20]). At each time step $k$ for which any voice exhibits a change point, we make an entry in **e** for every voice; we refer to $\mathbf{e}_k$ as a frame. This requires us to augment our alphabet of duration symbols $D$ with a special continuation symbol that indicates no change (comparable to the onset bits in the encoding **x**). An advantage of this representation over sequential or raster representations is that more history can be encoded with shorter sequences.

For a fixed voice $v$, let $\tilde{\mathbf{r}} \equiv \mathbf{e}_{:,v}$ be a single-voice slice of the score history. Observe that $\tilde{\mathbf{r}} \neq \mathbf{r}$, where **r** is the run-length encoding used for the single-voice task. The slices $\tilde{\mathbf{r}}$ are spaced out with aforementioned continuation symbols where there are change points in other voices. With the single-voice encoding **r**, simple linear filters can be learned that are sensitive to particular rhythmic sequences: e.g. groups of four eighth notes, or three triplet-quarter notes. This is not the case for $\tilde{\mathbf{r}}$; rhythmic patterns can be somewhat-arbitrarily broken up by continuation symbols.

These observations might lead us to consider raster encodings for multi-voice history, which restore the effectiveness of simple linear filters at the cost of increasing the dimensionality of the history encoding. We find that recurrent networks for the single-voice task are unhampered when retrained on $\tilde{\mathbf{r}}$: compare experiments 21 and 22 in Table 3. Performance falls slightly when learning on $\tilde{\mathbf{r}}$, but this is to be expected because history interspersed with continuations is effectively a shorter-length history.

For both the single-voice and multi-voice tasks, we truncate the history at a fixed number of frames prior to the prediction time. We explore several history lengths in the experiments and observe diminishing improvement in quantitative results for windows beyond the range of 10-20 frames of **e**: see experiment group (1,2,6,7) in Table 4.

### 5.2 Single-voice models

Using factorization (4), we explore fully connected, convolutional, and recurrent models for learning the conditional distributions $q(\mathbf{r}_{k,0}|\mathbf{r}_{1:k})$ over note-values and $q(\mathbf{r}_{k,1,n}|\mathbf{r}_{1:k}, \mathbf{r}_{k,0}, \mathbf{r}_{k,1,1:p})$ over pitches. We build separate models to estimate $\mathbf{r}_{k,0}$ and $\mathbf{r}_{k,1,p}$, with respective losses $\text{Loss}_t$ and $\text{Loss}_n$. In the remainder of this section, we consider opportunities to exploit structure in music by sharing weights in our models. Quantitative results for single-voice models are summarized in Table 3, with additional details available in Appendix A.

**Autoregressive modeling.** To build a generative model over conditionally stationary sequential data, it often makes sense to make the autoregressive assumption $q(\mathbf{r}_k|\mathbf{r}_{1:k}) = q(\mathbf{r}_{k'}|\mathbf{r}_{1:k'})$ for all $k, k' \in \mathbb{N}$. We can then learn a single conditional approximation $\hat{q}(\mathbf{r}_k|\mathbf{r}_{1:k})$ and share model parameters across all time translations.

Scores are not quite conditionally stationary; the distribution of notes and rhythms varies substantially depending on the sub-position within a beat. To address this, we follow the lead of [13] and [7] and augment our history tensor with a one-hot location feature vector $\ell$ that indicates the subdivision of the beat for which we are presently making predictions. [5] Compare the loss of duration models ($\text{Loss}_t$) with and without these features in experiment pairs (3,4), (6,7), (10,11), (12,13), and (15,16).

**Relative pitch.** We can perform a similar weight-sharing scheme with pitches as we did with time. Instead of building an individual predictor for each pitch conditioned on the notes in the history tensor, we adopt an idea proposed in [13]: build a single predictor that conditions on a shifted version of the history tensor centered around the note we want to predict. Convolving this predictor over the pitch axis of the history tensor lets us make a prediction at each note location, as visualized by Figure 1.

As with time, the distribution over notes is not quite conditionally stationary. For example, a truly relative predictor would generate notes uniformly across the note-class axis, whereas the actual distribution of notes concentrates around middle C. Therefore we augment our history tensor with a one-hot feature vector $\mathbf{1}_p$ that indicates the pitch $p$ for which we are making a prediction. This allows us to take full advantage of all available information when making a prediction, while borrowing strength from shared harmonic patterns in different keys or octaves. We compare absolute pitch-indexed classifiers ($\mathbf{lin}_p$) to a single, relative pitch classifier (**lin**) in Table 3: compare the loss of pitch models ($\text{Loss}_p$) in experiment groups (2,3,4), (5,6,7), (8,9,10), (11,12,13), and (15,16).

Relative pitch models serve a similar purpose to key-signature normalization [17] or data augmentation via transposition [7]. Building this invariance into the model

[5] Location can always be computed from a full history. But we truncate the history, so this information is lost unless it is explicitly reintroduced.

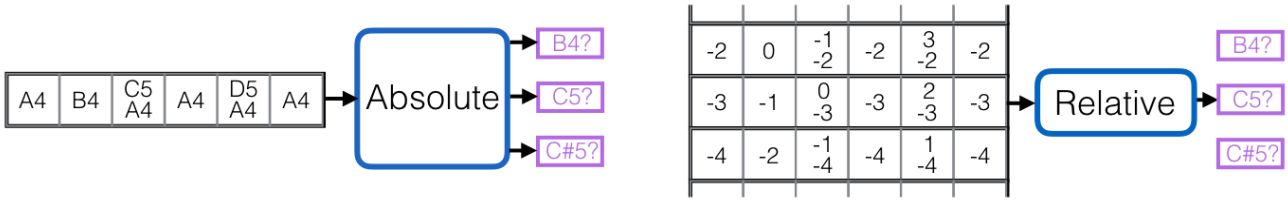


**Figure 1**. Left: an absolute pitch predictor learns individual classifiers for each pitch-class. Right: a relative pitch predictor learns a single classifier and translates the data along the frequency axis to center it around the pitch to be predicted. Whereas the absolute predictor decides whether C5 is on given the previous note was A4, the relative predictor decides whether the note under consideration is on given the previous note was 3 steps below it.

offers an alternative approach, avoiding data preprocessing or the introduction of hyper-parameters. We find that training with transpositions in the range $\pm 5$ semi-tones yields no performance increase for relative pitch models.

**Pitch embeddings.** Borrowing the concept of a word embedding from natural language processing, we consider learned embeddings $\mathbf{c}$ of the pitch vectors $\mathbf{r}_{k,1}$ ($\mathbf{e}_{k,v,1}$ for the multi-voice models). For recurrent models, we do not see performance benefits to learning these embeddings: compare experiments 20 and 21 in Table 3. However, we do find that we can learn compact embeddings (16 dimensions for the experiments presented in this paper) without sacrificing performance, and using these embeddings reduces computational cost. We also find that using a 12 dimensional fixed embedding of pitches $\mathbf{f}$, in which we quotient each pitch class by octave, reduces overfitting for the rhythmic model while preserving predictive accuracy.

### 5.3 Multi-voice models

Using the factorization (5), we now explore ways to capture correlations between the voices and model the conditional distributions $q(\mathbf{s}_{k,0}|\mathbf{s}_{1:k})$ over note-values and $q(\mathbf{s}_{k,1,p}|\mathbf{s}_{1:k}, \mathbf{s}_{k,0}, \mathbf{s}_{k,1,1:p})$ over notes. We build separate models to estimate $\mathbf{r}_{k,0}$ and $\mathbf{r}_{k,1,p}$, with losses $\text{Loss}_t$ and $\text{Loss}_p$ in Table 4 respectively. The same structural observations that we made about scores for the single-voice models apply to multi-voice modeling; all multi-voice models considered in this paper use the three weight-sharing schemes considered for single-voice models. We explore an additional weight-sharing opportunity below for the multi-voice task: voice decomposition.

The effectiveness of recurrent models for the single-voice modeling task, and the representational considerations in Section 5.1, motivate us to consider extensions of the recurrent architecture to capture structure in the multi-voice setting. One natural extension of the standard recurrent neural network to model multiple, concurrent voices is a hierarchical architecture, illustrated in Figure 2:

$$\begin{aligned} h_{k,v}(\mathbf{e}) &\equiv \mathbf{a}\left(W_v^\top h_{k-1,v}(\mathbf{e}) + W_e^\top \mathbf{c}\left(\mathbf{e}_{k,v}\right)\right), \\ g_k(\mathbf{e}) &\equiv \mathbf{a}\left(W_g^\top g_{k-1}(\mathbf{e}) + W_{hv}^\top \sum_u h_{k,u}(\mathbf{e})\right). \end{aligned} \tag{7}$$

The first equation is a standard recurrent network that builds a state estimate $h_{k,v}$ of a voice $v$ at time index $k$ based on transition weights $W_v$, an input embedding $\mathbf{c}$, input weights $W_e$, and non-linear activation $\mathbf{a}$ (we use a ReLU activation). We integrate the state of each voice (weights $W_{hv}$) into a global state $g_k$ given the previous global state $g_{k-1}$ (weights $W_g$). Because voice order is arbitrary in our dataset, we sum (i.e. pool) over their states before feeding them into the global network. At each time $k$, we use the learned state of each voice together with the global state to make a note-value prediction: $\hat{\mathbf{s}}_{k,0} = \mathbf{lin}(h_{k,\beta_k}(\mathbf{e}), g_k(\mathbf{e}))$, where $\mathbf{lin}$ is a log-linear classifier. We make pitch predictions $\mathbf{s}_{k,1,p} \in \{0, 1\}$ using the same architecture. We learn a single, relative-pitch classifier for $\mathbf{s}_{k,1,p} \in \{0, 1\}$ in all multi-voice experiments

| # | History | Arch | Loc? | Relative? | Pitch? | Embed? | Loss |
|---|---|---|---|---|---|---|---|
| 1 | $\mathbf{r}_{(1)}$ | bias | no | no | no | no | 10.07 |
| 2 | $\mathbf{r}_{(1)}$ | linear | no | no | no | no | 8.05 |
| 3 | $\mathbf{r}_{(1)}$ | linear | no | yes | no | no | 6.29 |
| 4 | $\mathbf{r}_{(1)}$ | linear | yes | yes | yes | no | 6.12 |
| 5 | $\mathbf{r}_{(1)}$ | fc | no | no | no | no | 5.92 |
| 6 | $\mathbf{r}_{(1)}$ | fc | no | yes | no | no | 6.05 |
| 7 | $\mathbf{r}_{(1)}$ | fc | yes | yes | yes | no | 5.70 |
| 8 | $\mathbf{r}_{(5)}$ | linear | no | no | no | no | 7.91 |
| 9 | $\mathbf{r}_{(5)}$ | linear | no | yes | no | no | 5.76 |
| 10 | $\mathbf{r}_{(5)}$ | linear | yes | yes | yes | no | 5.63 |
| 11 | $\mathbf{r}_{(5)}$ | fc | no | no | no | no | 4.90 |
| 12 | $\mathbf{r}_{(5)}$ | fc | no | yes | no | no | 4.80 |
| 13 | $\mathbf{r}_{(5)}$ | fc | yes | yes | yes | no | 4.69 |
| 14 | $\mathbf{r}_{(5)}$ | fc | yes | yes | yes | yes | 4.63 |
| 15 | $\mathbf{r}_{(10)}$ | linear | no | yes | no | no | 7.88 |
| 16 | $\mathbf{r}_{(10)}$ | linear | yes | yes | yes | no | 5.53 |
| 17 | $\mathbf{r}_{(10)}$ | fc | yes | yes | yes | yes | 4.55 |
| 19 | $\mathbf{r}_{(10)}$ | cnn | yes | yes | yes | yes | 4.42 |
| 20 | $\mathbf{r}_{(10)}$ | rnn | yes | yes | yes | no | 4.37 |
| 21 | $\mathbf{r}_{(10)}$ | rnn | yes | yes | yes | yes | 4.36 |
| 22 | $\tilde{\mathbf{r}}_{(10)}$ | rnn | yes | yes | yes | yes | 4.52 |

**Table 3**. Single-voice results. We define $\mathbf{r}_{(m)} \equiv \mathbf{r}_{k-m:k}$ (a truncated history of length $m$); $\tilde{\mathbf{r}}_{(m)}$ is defined likewise, based on the alternate encoding $\tilde{\mathbf{r}}$ discussed in Section 5.1, Representation. The Relative flag indicates the use of a relative-pitch classifier, and the Loc, Pitch, and Embed flags indicate the use of location features, pitch features, and pitch embeddings, discussed in Section 5.2. For additional details of these experiments, see Appendix A.

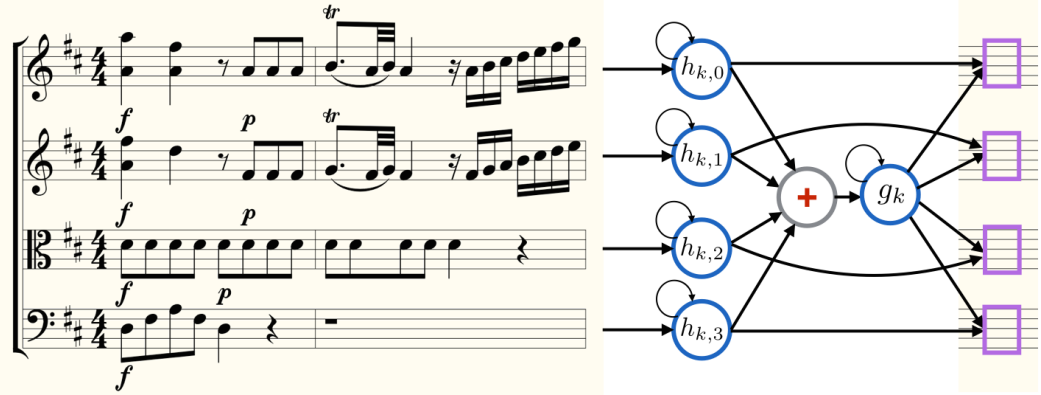


**Figure 2**. Coupled state estimation of Mozart's string quartet number 2 in D Major, K155, movement 1, from measure 1, rendered by the Verovio Humdrum Viewer. A recurrent network models the state $h_{k,v}$ of each voice $v$ at step $k$, based on the previous state $h_{k-1,v}$ and the current content of the voice. Another recurrent network models of the global state $g_k$ of the score at step $k$ based on the previous global state $g_{k-1}$ and a sum of the current states of each voice. Subsequent notes (purple) in each voice are predicted using features of the global state and the state of the relevant voice. See Equations 7 for a precise mathematical description of this model.

(section 5.2, Relative pitch). We do not share weights between the note-value and pitch models.

**Voice decomposition.** Decomposing a score into multiple voices presents us with an opportunity to share weights between voice models by learning a single set of weights $W_v$ in equation (7), rather than learning unique voice-indexed weights $W_{v_i}$ for each voice $v_i$. Indeed, because voice indices are arbitrary, the weights $W_{v_i}$ will converge to the same values for all $i$; sharing a single set of weights $W_v$ accelerates learning by enforcing this property. All score models presented in Table 4 share these weights.

| # | History (voice/global) | Architecture | Loss (total) | $\text{Loss}_t$ (time) | $\text{Loss}_n$ (notes) |
|---|---|---|---|---|---|
| 1 | 3 / 3 | hierarchical | 14.05 | 5.65 | 8.40 |
| 2 | 5 / 5 | hierarchical | 13.40 | 5.35 | 8.04 |
| 6 | 10 / 10 | hierarchical | 12.87 | 5.12 | 7.75 |
| 7 | 20 / 20 | hierarchical | 12.78 | 5.01 | 7.76 |
| 8 | 10 | independent | 18.63 | 6.56 | 12.08 |

**Table 4**. Multi-voice results. The "hierarchical" architecture is defined by equations (7). Voice and global history refer to the number of time steps used to construct the states $h_{k,v}$ and $g_k$ respectively. Experiment 8 is a baseline where the voice models are completely decoupled (equivalent to single-voice Experiment 22 in Table 5; the average number of voices per score is 4.12). Results are reported on non-piano test set data (see Appendix B for discussion of piano data). For additional experiments and ablations, see Appendix A.

## 6. CONCLUSION

To gain insight into the quality of samples from our models, we recruited twenty study participants to listen to a variety of audio clips, each synthesized from either a real composition or from sampled output of Experiment 6 in Table 4. For each clip, participants were asked to judge whether the clip was written by a computer or by a human composer, following a procedure comparable to [21]. The clips varied in length, from 10 frames of a sample **e** (2-4 seconds; the length of history conditioned on by the model) to 50 frames (10-20 seconds). Participants become more confident in their judgements of the longer clips, but even among the longest clips (around 20 seconds) participants often identified an artificial clip as a human composition. Results are presented in Table 5; see Appendix E for further study details.

| Clip Length | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|
| Average | 5.3 | 5.7 | 6.6 | 6.7 | 6.8 |

**Table 5**. Qualitative evaluation of the 10-frame hierarchical model: Experiment 6 in Table 4. Twenty participant were asked to judge 50 audio clips each, with lengths varying from 10 to 50 frames. The scores indicate participants' average correct discriminations out of 10: 5.0 would indicate random guessing; 10.0 would indicate perfect discrimination.

These results superficially suggest that we have done well in modeling the short-term structure of the dataset (we make no claims to have captured long-term structure; indeed, the truncated history input to our models precludes this). But it is not clear that humans are good–or should be good–at recognizing plausible local structures in music without context. See [14, 22] for criticism of musical Turing tests like the one presented here. It is also unclear how to use such studies to make fine-grained comparisons between models (as we have done quantitatively throughout this paper). It is not even clear how to prompt a user to discriminate between such models. Therefore we re-emphasize the interpretation of this qualitative evaluation, proposed in Section 1, as a perceptual grounding of the quantitative evaluation considered throughout this work.

## 7. ACKNOWLEDGEMENTS

We thank Lydia Hamessley and Sreeram Kannan for sharing valuable insights. This work was supported by NSF Grants DGE-1256082, CCF-1740551, the Washington Research Foundation for innovation in Data-intensive Discovery, and the CIFAR program "Learning in Machines and Brains." We also thank NVIDIA for their donation of a GPU.

## 8. REFERENCES

[1] Moray Allan and Christopher K. I. Williams. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *Advances in Neural Information Processing Systems*, 2006.

[2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *International Conference on Machine Learning*, 2012.

[3] Darrell Conklin. Music generation from statistical models. In *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 30–35, 2003.

[4] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Language modeling with longer-term dependency. 2018.

[5] Kemal Ebcioğlu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 1988.

[6] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 2010.

[7] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation. *International Conference on Machine Learning*, 2017.

[8] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the maestro dataset. *arXiv preprint arXiv:1810.12247*, 2018.

[9] Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. A functional taxonomy of music generation systems. *ACM Computing Surveys (CSUR)*, 50(5):69, 2017.

[10] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. *International Society for Music Information Retrieval Conference*, 2017.

[11] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. 2019.

[12] Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. Tuning recurrent neural networks with reinforcement learning. *International Conference on Learning Representations Workshop*, 2017.

[13] Daniel D. Johnson. Generating polyphonic music using tied parallel networks. *International Conference on Evolutionary and Biologically Inspired Music and Art*, 2017.

[14] Anna Jordanous. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation*, 4(3):246–279, 2012.

[15] Teuvo Kohonen. A self-learning musical grammar, or 'associative memory of the second kind'. *International Joint Conference on Neural Networks*, 1989.

[16] Victor Lavrenko and Jeremy Pickens. Polyphonic music modeling with random fields. *ACM International Conference on Multimedia*, 2003.

[17] Feynman Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. Automatic stylistic composition of bach chorales with deep lstm. *International Society for Music Information Retrieval Conference*, 2017.

[18] Michael C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 1994.

[19] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[20] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: Learning expressive musical performance. *arXiv preprint arXiv:1808.03715*, 2018.

[21] Marcus Pearce and Geraint Wiggins. Towards a framework for the evaluation of machine compositions. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 22–32, 2001.

[22] Marcus T Pearce and Geraint A Wiggins. Evaluating cognitive models of musical composition. In *Proceedings of the 4th international joint workshop on computational creativity*, pages 73–80. Goldsmiths, University of London, 2007.

[23] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. *arXiv preprint arXiv:1803.05428*, 2018.

[24] Craig Stuart Sapp. Online database of scores in the humdrum file format. *International Society for Music Information Retrieval Conference*, 2005.

[25] Roger N. Shepard. Geometrical approximations to the structure of musical pitch. *Psychological Review*, 1982.

[26] Bob L. Sturm, Joao Felipe Santos, Oded Ben-Tal, and Iryna Korshunova. Music transcription modelling and composition using deep learning. *Conference on Computer Simulation of Musical Creativity*, 2016.

[27] John Thickstun, Zaid Harchaoui, and Sham M. Kakade. Learning features of music from scratch. In *International Conference on Learning Representations (ICLR)*, 2017.

[28] Peter M. Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 1989.

[29] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *SSW*, 125, 2016.

[30] Raunaq Vohra, Kratarth Goel, and J. K. Sahoo. Modeling temporal dependencies in data using a dbn-lstm. *IEEE International Conference Data Science and Advanced Analytics*, 2015.