Proceedings of Machine Learning Research 133:253–274, 2021 NeurIPS 2020 Competition and Demonstration Track

Musical Speech: A Transformer-based Composition Tool

Jason d'Eon* JNDEON@DAL.CA
Sri Harsha Dumpala* SRIHARSHA.D@DAL.CA
Chandramouli Shama Sastry* CSSASTRY@DAL.CA
Dalhousie University, Vector Institute

Dani Oore DOORE@MUN.CA
IICSI, Memorial University of Newfoundland

Sageev Oore SAGEEV@VECTORINSTITUTE.AI
Dalhousie University, Vector Institute

Editors: Hugo Jair Escalante and Katja Hofmann

Abstract

In this paper, we propose a new compositional tool that will generate a musical outline of speech recorded/provided by the user for use as a musical building block in their compositions. The tool allows any user to use their own speech to generate musical material, while still being able to hear the direct connection between their recorded speech and the resulting music. The tool is built on our proposed pipeline. This pipeline begins with speech-based signal processing, after which some simple musical heuristics are applied, and finally these pre-processed signals are passed through Transformer models trained on new musical tasks. We illustrate the effectiveness of our pipeline – which does not require a paired dataset for training – through examples of music created by musicians making use of our tool.

Keywords: Speech processing, musical notes, transformer networks, denoising autoencoder.

1. Introduction

Among the extensive recent works applying machine learning to the generation of music and sound (Dhariwal et al., 2020; Dieleman et al., 2018; Technologies, 2020; Payne, 2019; Engel et al., 2020; Oore et al., 2018; Vasquez and Lewis, 2019), an interesting and important direction are those works that aim to provide musicians with compositional tools. One approach in this domain is tools that "generate material", perhaps in some way based on initial musical ideas or directions set by the user, e.g. Huang et al. (2017, 2018a); Donahue et al. (2019); Roberts et al. (2018); Meade et al. (2019). Of particular interest to us are vocal-based tools that generate musical ideas conditioned on a recording of the user's voice. The concept of converting speech into music has a long musical history (see Section 2.4 for a detailed discussion), but no techniques to automate parts of the process have been described in the literature as far as we know. It is an important compositional technique that is challenging to do even for highly skilled musicians, yet the results have often been very effective. While we are not aware of studies that explore why, in our experience, musicians who have a skilled enough ear (and/or perfect pitch) that they are

* Equal Contribution

able to accurately transcribe musical excerpts, still find it challenging and time-consuming to "transcribe" speech to music.

There is another, quite different reason we are interested in converting speech to music: separating prosody and timbre from content provides an informative sonification of paralinguistic characteristics of speech (e.g. Dumpala et al. (2020)), which are informative distinctly from the semantic content. For example, recent work uses paralinguistic characteristics of speech to predict information about the mental health of the speaker (Dumpala et al., 2021; Cheng et al., 2020). In another example, musically-driven analyses of rhythmic aspects of speech (obtained by meticulous manual extraction of musical characteristics from political speeches) have been used to discover rhythmic motifs and examine their role in socio-political contexts (Oore, 2018).

We thus propose developing machine learning tools that allow a user to generate musical source material by translating their speech into melodic fragments. To achieve this we combine speech-based signal processing, musical heuristics, and a Transformer model. We needed to define new musical tasks in order to train this Transformer model, that would represent the requirements of our context. Specifically, the raw outputs of the speech processing (e.g. extraction of formant parameters, as described below) result in a barrage of quickly changing musical pitches, and we needed a system that could find a medium between this raw output, and pure generative model outputs.

We will outline the approach in Section 3, provide details in the next sections, and in Section 8 present some analysis of the results and, crucially, examples of compositional excerpts created from short audio recordings of speech using our system. First we give background information on the three main elements of this system: speech processing, conditional music generation, and the historical conversion of speech into music.

2. Background and Related Work

2.1. Speech Processing

Speech and music are two distinct aural phenomena perceived by the same human auditory mechanism, and often treated separately (e.g. whether for analysis or generation). A variety of works have explored some of the connections between these phenomena (Hausen et al., 2013; Ding et al., 2017). Ding et al. (2017) show that the rhythmic structure is a fundamental feature of both speech and music. Both domains involve sequences of events (such as syllables or notes) which have systematic patterns of timing, accent, and grouping (Patel, 2010), but the rhythmic pattern of speech is distinct from that of music (Ding et al., 2017). In this paper, we slightly adjust rhythmic patterns of speech to map them to rhythmic patterns of music. Further, it was observed by linguists that human speech has a temporal rhythm that can be characterized by placing a perceptual "beat" around successive syllables (Port, 2003). In this work, we exploit this observation to obtain an intermediate representation of the speech signal by considering the acoustic features around regions with high intensity or around syllable nuclei positions in speech.

2.2. Music Generation

In recent years, considerable work has been done in the application of machine learning in musical contexts. Such musical generation and processing is generally approached at one of two levels: (1) at the raw audio level, where a waveform is represented at a sample rate on the order of 16 KHz, and (2) at the note-level, where the gaps between notes of a musical performance can be represented with a sample rate of roughly 50Hz (that is, there are usually no more than 16 notes per second, usually much fewer, but for reasonable fidelity, the spaces between them need to be represented with a resolution of at least 20ms or better). The note-level approach—and the one that we take here—is considered symbolic (i.e. each "note" is one element in a finite set of notes, rather than a continuous waveform), and the representation is usually based on the MIDI-format. For purposes of this paper, this simply means each note is a symbol, and each durational event (e.g. length of a note, distance between onsets of two notes) has been discretized and can thus be tokenized as well.

In general, generating symbolic music (Huang et al., 2018a; Payne, 2019; Oore et al., 2018) in the form of MIDI is relatively easier than directly generating raw audio of music (Dhariwal et al., 2020; Dieleman et al., 2018; Vasquez and Lewis, 2019) as generating raw audio involves maintaining coherence over several thousand samples even for a clip of few milliseconds. We use speech processing techniques to move from the waveform domain of raw speech to the symbolic domain of note-based representation, and then build upon previous works Huang et al. (2018a); Oore et al. (2018) to transform one (speech-derived) note sequence into another (musical) one.

Recent generation systems have tended to focus on providing the user with some degree of control over the generation, either through conditioning signals or through priming. For example, Meade et al. (2019) and Louie et al. (2020) both explore a variety of possible ways to condition a generated musical sequence, from composer to histograms of loudness (i.e. MIDI velocities).

2.3. ML-based Tools for Musicians

As more work in machine learning is being done with the goal of providing musicians with tools for supporting music creation, more attention is being paid to how these proposed tools are being used (Huang et al., 2020). A motivation for much of the work on controlling musical generation has been that one of the underlying recent goals is to build tools for musicians and producers (of varying skill levels). That is, automating music generation is not an end in itself, but rather it is intended to provide people with new processes to make music themselves. *Piano genie* (Donahue et al., 2019) is an extremely entertaining system that allows anybody to play on a small toy keyboard and have it turn into impressive piano music that follows rhythmic and other musical features of the original input. Castro (Castro, 2019) describes and very effectively demonstrates a framework for structured musical improvisation with trained generative models. Others have also explored musical improvisation with a generative model as well, e.g. Bretan et al. (2017); Roberts et al. (2016). Following earlier works on timbral conversion (Huang et al., 2018b; Mor et al., 2018), the DDSP project (Engel et al., 2020) provides the user with a sophisticated tool for transforming timbral and other characteristics of an existing audio recording. For example,

the user can record themselves singing and then use DDSP to process that audio keep the pitch and convert it into the sound of a trumpet.

Our current system is designed in this spirit as well: it is not meant to create a piece of music, but rather to generate musical material, derived from speech and adjusted to fit a musical language model, that a composer can then work with, as we will describe in Section 8.2.

2.4. Speech to Music

There is a long musical tradition of looking to speech itself as a source of inspiration. the great Brazilian musician Hermeto Pascoal called the technique *Som da Aura* (Boukas, 2013), and it has been the basis of many songs, videos, and even entire albums (Spearin and Band, 2009), in musical contexts of jazz, classical (Reich, 1988), African (Beier, 1954) and other traditions. Oore (2021) compiled a playlist of over 300 musical videos in response to Donald Trump, many of them using his idiotsyncratic speech patterns as source material for musical compositions. While getting speech samples is easy, "finding" coherent melodic structure underlying speech is a challenging task even for experienced musicians. One system providing speech-to-melody functionality is a commercial audio workstation software AG (2019), but the automated results of such conversion can be musically complex and perhaps unintuitive both rhythmically and harmonically, since the voice often moves continuously through many notes. That is, when a novice user applies such a conversion, it may be unclear to them what to do with the resulting melody. When musicians do this conversion effectively (by ear), they combine the important pitches they hear in the voice *together with extensive musical skill*, perhaps analogously to how speech recognition systems incorporate language models.

One of our contributions here is not just in the automated conversion from speech to pitch, but in the multi-step pipeline to do so: the use of a Transformer allows us to incorporate a musical language model in this way, and the sparsification can work because the musical transformer has been trained to effectively fill in the gaps between such constraints. The Transformer's role here is critical, because it "makes musical sense" of the provided constraints, i.e. it chooses notes and rhythms such that those constraints feel musically natural. For example, we have observed the system adding in a note just a fraction of a second before the melody begins, in such a way as to give the resulting filled-in excerpt a much clearer rhythmic structure than it had by the constraints alone[1].

Our approach is unique in its use of Transformers, conditioned on sparse musical note representation of speech, to generate a complete sequence of musical notes following the input speech pattern. The result is a new compositional tool that allows any user—regardless of musical training—to use their own speech to generate musical, satisfying melodies, while still being able to hear the direct connection between their recorded speech and the resulting music.

2.5. Transformers

Transformer is a sequence-to-sequence model introduced by Vaswani et al. (2017) as an improvement over recurrent neural networks. Typically, the sequence-to-sequence model

1. Link to demo video

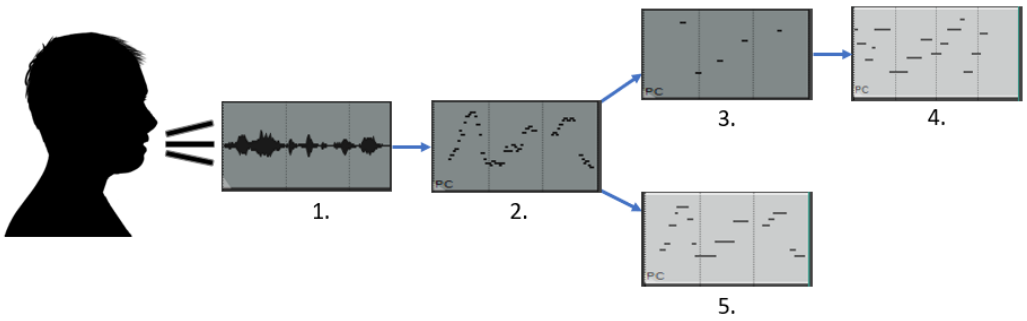

Figure 1: The speech to music conversion pipeline. From the speech audio signal (1), we extract F_0, F_1, F_2, or F_3 formant values to obtain a symbolic audio representation (2). From this, we sparsify the sequence according to a heuristic (3) and apply the gap-filling Transformer (4), or alternatively, we directly apply the denoising Transformer (5).

contains an encoder and a decoder, each comprising of a set of self-attention layers. The encoder processes an input sequence and feeds into the decoder, which autoregressively generates the output sequence. In our case, the input and output are MIDI sequences.

Besides the Music Transformer (Huang et al., 2018a), our training setup and the Transformer architecture also draws inspiration from the Masked Language Modeling tasks and/or the architecture used in training Transformer models for natural-language understanding and generation tasks. In the Masked Language Modeling (MLM) task, the model is asked to predict a part of the text that is masked out and replaced with mask tokens. BERT (Devlin et al., 2019) originally demonstrated the potential of training a Transformer on the MLM task and has directly/indirectly inspired a line of works including T5 (Raffel et al., 2020), XLNet (Yang et al., 2019), and BART (Lewis et al., 2020). Architecture-wise, BERT is an encoder-only Transformer and is not suitable for sequence generation out-of-the-box (see (Rothe et al., 2020)); as described below, we use an encoder-decoder architecture and replace a contiguous sequence of tokens to be masked in our input sequence with a single mask token – unlike BERT, which has to replace each word in a span with a mask token on its own, a direct consequence of using an encoder-only architecture – and train the decoder to predict the masked portions following architectures like T5 (Raffel et al., 2020), BART (Lewis et al., 2020), and SpanBERT (Joshi et al., 2020).

3. Contributions

In this work, we demonstrate a system that will generate a musical line (melody) that follows the prosody of speech recorded or provided by the user. An overview of the system pipeline is shown in Figure 1.

Our contributions include the following:

- a functional pipeline for conversion from speech to music, despite that no such parallel dataset exists (Figure 1 and Sections 4-8)

- definition of a new set of musical composition tasks for denoising and gap-filling
- translation of these tasks into Transformer training
- exploratory analysis of how the translation adjusts characteristics of the raw musical signal obtained directly from speech (Section 8.1)
- an interactive tool that can be used online by anyone (Section 7)
- demonstration and discussion of how this tool can be used by expert musicians for creating musical excerpts (Section 8.2).

4. Pre-Processing Speech Data

4.1. Feature Extraction

Spectral and prosodic features are commonly used to represent speech signal characteristics. The features we consider in this work are the following:

Fundamental frequency F_0 refers to the rate of the vocal fold vibration. Several algorithms were proposed to estimate the F_0 contour of speech (De Cheveigné and Kawahara, 2002; Yegnanarayana and Murty, 2009; Morise et al., 2009). In this work, we use the DIO algorithm (Morise et al., 2009, 2016) to estimate F_0, based on period detection of the vocal fold vibration. DIO is a fast and reliable algorithm for estimating F_0 for speech and singing voice.

Formant frequencies F_1, F_2, and F_3 refer to resonances of the vocal tract system during production of speech (Epps et al., 1997). Formant estimation from speech is a challenging problem, and algorithms with varying complexity have been proposed for this task (Boersma, 2001; Deng et al., 2006; Durrieu and Thiran, 2013). In this work, we use the approach proposed in Boersma (2001), which is based on a simple linear prediction analysis (Makhoul, 1975; Rabiner and Juang, 1993).

Loudness contour of the speech signal is computed as the mean of the perceptually weighted (A-weighting) short-term power spectrum of speech (Mermelstein, 1975). In this work, we consider the loudness contour to select the regions of interest in speech using two different sparsification techniques.

4.2. Sparsification techniques

The sparsification step is used to select the regions of interest in speech which guide the transformer to generate a musical line (melody) that follows the characteristics of speech provided by the user. In this paper, we follow two approaches to select the regions of interest in the input speech signal. 1) a heuristic-based approach and 2) a syllable-nuclei-based approach.

Heuristic-based approach: The steps in the heuristic-based approach for sparsification of speech are as follows:

1. Compute the short-term loudness contour (Mermelstein, 1975) of the speech signal. The short-term loudness contour is extracted using a window of length 50 ms and a frame-shift/hop-size of 20 ms.

2. Smoothen the loudness contour using a moving-average smoothing technique.

3. Select only those values in the loudness contour which are higher than a pre-defined threshold and discard other values. The pre-defined threshold is set based on empirical analysis of the loudness contours obtained from a set of speech signals.

4. Consider only the values of F_0, F_1, F_2 and F_3 in the regions of interest as obtained in Step-3 for further processing.

Syllable-nuclei-based sparsification: In this approach we consider syllable nuclei (Ladefoged and Johnson, 2014) positions in the speech signal as the regions of interest. Segmenting the loudness contour recursively using the convex-hull algorithm provides a syllable-level segmentation of speech. The peak in the loudness contour in each segment refers to the *syllable nuclei* (Zhang and Glass, 2009). In this work we consider the algorithm in De Jong and Wempe (2009) which uses a combination of intensity (Pfitzinger, 1999) and voicedness (Pfau and Ruske, 1998) to detect the syllable nuclei locations in speech (Described further in Appendix B).

4.3. Sparsification levels

We allow three levels of sparsification i.e., low, medium and high, for each of the two sparsification techniques discussed above. In low-level sparsification, most of the values of F_0, F_1, F_2 and F_3 are retained whereas in the high-level sparsification, very few of the F_0, F_1, F_2 and F_3 values are retained. For heuristic-based sparsification, the threshold is varied depending on the level of sparsification. For syllable-nuclei-based sparsification, the context (number of frames selected) around each syllable nuclei is varied depending on the level of sparsification. For instance in heuristic-based approach, for lower level of sparsification, a lower threshold value is considered. Similarly for the syllable-nuclei-based sparsification, two values on each side of the syllable nuclei (along with the value at syllable nuclei) are retained.

Figure 2 shows the retained F_0 in the regions of interest as obtained by the two sparsification approaches. For Figure 2, we considered medium-level of sparsification for both approaches.

5. MAESTRO Dataset

As mentioned, we used the MIDI format to obtain tokenized representations of music, which captures note pitch (integer in [0-127]), velocity, start time, and end time. All models were trained using the MIDI component of the MAESTRO V2.0.0 dataset (Hawthorne et al., 2019). This dataset consists of approximately 200 hours of professional classical piano performances. Because the extracted speech sequences from voice recordings are monophonic (single-pitched at any given time), our generative models were entirely focused on monophonic generation. In order to use MAESTRO, which consists of polyphonic files, we first pre-processed by extracting the highest note being played at any given time, in a so-called "skyline" heuristic (see Figure 3 for an example). Note that this method is far from being proper melody extraction, which in itself, is a challenging problem in the music domain. Rather, this method provides us with harmonically reasonable sequences of monophonic music.

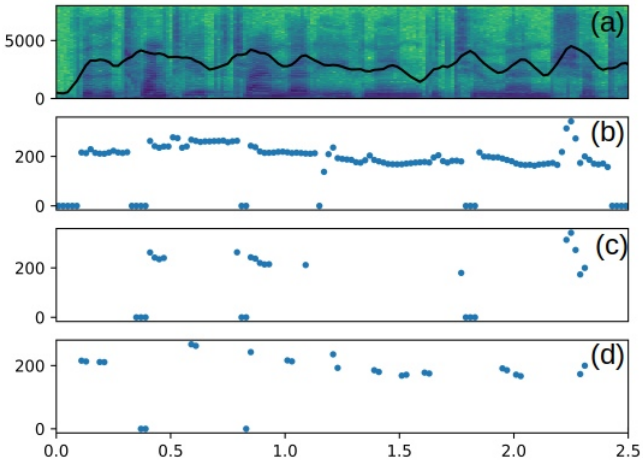

Figure 2: Figure depicting the sparsification techniques. (a) spectrogram of input speech signal. The black line shows the smoothed loudness contour, (b) F_0 contour of the speech signal, (c) F_0 values retained after applying heuristic-based sparsification and (d) F_0 values retained after applying syllable-nuclei-based sparsification

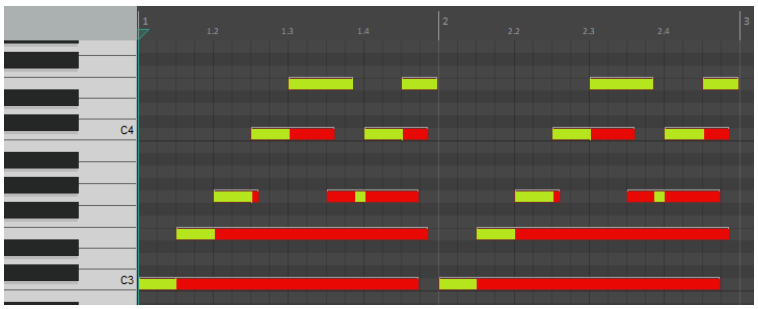

Figure 3: An example of the skyline heuristic on a polyphonic file. Green indicates the monophonic sequence that is extracted from this file.

To provide numerical inputs to the models, we represented monophonic sequences as a time-series of held pitches with a time-step of 20 ms, which is around the level of human imperceptibility. For example, an input of:

$$[S, 40, 40, 40, 40, 40, 42, 42, 42, 42, 42, 0, 0, 0, 0, 0, E],$$

indicates the 40th piano note being held for 0.1 s, followed by the 42nd note being held for 0.1 s. S and E are the start and end tokens respectively, and 0 represents silence. Due to the space complexity limitations of Transformer, we restricted inputs to a length of 502 (10 s sequence with start and end tokens), and we prepared 10 s samples from MAESTRO accordingly.

6. Seq2seq Methods

6.1. Overview

We trained sequence-to-sequence models that would modify raw speech sequences by injecting musical structure learned from MAESTRO. We settled on two approaches:

1. Apply one of the two sparsification techniques to decrease the density of the raw speech sequence and fill the gaps back in with a "gap-filling" Transformer model trained on MAESTRO.

2. Directly apply a denoising Transformer model to the raw speech sequences that is trained to reduce the chromaticism of the sequence in a way that reveals the underlying musical structure.

6.2. Gap-filling Transformer

The gap-filling model consists of an encoder-decoder architecture (Vaswani et al., 2017). The inputs to the encoder and decoder are both initially passed through a 512-dimensional embedding layer which is learned during training. Each encoder and decoder layer has a self-attention component and a feed-forward component. The encoder and decoder both have 6 layers, each with 8 attention heads and 1024-dimensional feed-forward layers. Observing that the note sequences consist of repeated tokens, we experimented with predicting the token-pitch and its token-count in the same unrolling step; empirically, we found this scheme to work better than predicting the same token-pitch over successive unrolling steps. Therefore, on a given encoder-decoder input pair, the output of the decoder is passed to both a linear pitch predictor, and a linear-sigmoid token count predictor. During training, counts are scaled from $[1, 500]$ to $[0, 1]$, to suit the sigmoid function range.

In addition, we also employed the use of relative position-based attention, following the method of Huang et al. (2018a), which has been shown to perform better than sinusoidal positional encoding at capturing long-term structure.

We designed a task where we mask segments of a MAESTRO sequence and learn to map masked sequences to the original sequence. Random masks are created by the following procedure:

1. Construct a multiset M from the set of token lengths $S = \{25, 50, \ldots, 150\}$ such that $\sum_{i \in M} i = 150$.

2. For every element $i \in M$, i consecutive tokens are replaced with the <gap> token in the input. This is done by uniformly choosing a continuous segment of length i that does not contain any <gap> token.

The decoder is trained to predict what the gap tokens should be and therefore, the loss is evaluated only over the predictions that correspond to the <gap> tokens: the pitch prediction is evaluated with a negative log-likelihood loss, while the token count prediction is evaluated with mean squared error loss. Note that this is reminiscent of the CocoNet project (Huang et al., 2017) in which polyphonic musical scores are "filled in". Additionally, we augment the dataset by performing pitch transposition between -5 and $+5$ semitones during sampling. Adam optimizer was used, following the learning rate schedule described in Vaswani et al. (2017). A dropout value of 0.1 was used on all layers during training. The model was trained with a batch size of 8 for approximately $230,000$ iterations on 4 Nvidia Tesla P100 GPUs.

6.3. Denoising Transformer

The denoising model also follows an encoder-decoder structure, almost identical to the gap-filling model. The embedding dimension is 128, followed by 2 encoder and decoder layers, with 1024-dimensional feed-forward sub-layers. Each attention module has 2 attention heads. The decoder output is passed to a linear layer to do pitch prediction only. Unlike the gap-filling model, pitch is predicted token by token.

The training task was to map noisy inputs to their original state. Noisy inputs are created by the following procedure:

1. Generate a sequence of random variables from a normal distribution of zero mean and unit variance, with a length equal to the input.

2. Round the random variables to the closest integer.

3. Add the sequence of random variables to the input sequence under the condition that 0 tokens remain, tokens in the range $[1, 88]$ stay in this range, and end tokens remain the same.

Pitch prediction is evaluated under a negative log-likelihood loss. The optimizer and learning rate schedule are identical to Vaswani et al. (2017). A dropout value of 0.1 was used on all layers during training. The model was trained with a batch size of 32 for approximately $260,000$ iterations on a single Nvidia Tesla P100.

6.4. Inference

As described above, the MIDI representation of a melody consists of a sequence of note-pitches along with their velocities, and start and end times. The Transformer models determine the note-pitches and their durations in the output sequence, while their velocities

are determined from the speech loudness by mapping the loudness to an integer in the closed-interval [0,127]. The inference is done depending on the Transformer model as described below:

- Gap-filling Transformer: Either one of the F_0, F_1, F_2 or F_3 sequences, extracted from the input speech, is sparsified using any one of the techniques described in Section 4.2. The parts of the sequence chosen to be discarded as a part of the sparsification process are replaced with <gap> tokens. The pitch is randomly chosen from the softmax output of the model, while the length is deterministically obtained by mapping the sigmoid output back to [1,500]. The Transformer model is then continuously unrolled unless all of the gap tokens are filled; if the token-count predicted in the latest unrolling step does not match with the remaining number of consecutive gap tokens, the extra tokens are discarded.

- Denoising Transformer: Either one of the F_0, F_1, F_2 or F_3 sequences, extracted from the input speech, is provided as input to the denoising Transformer, and the resulting denoised output – of the same length as that of the input – is used to infer the pitches and durations of the refined MIDI sample.

7. User Interface

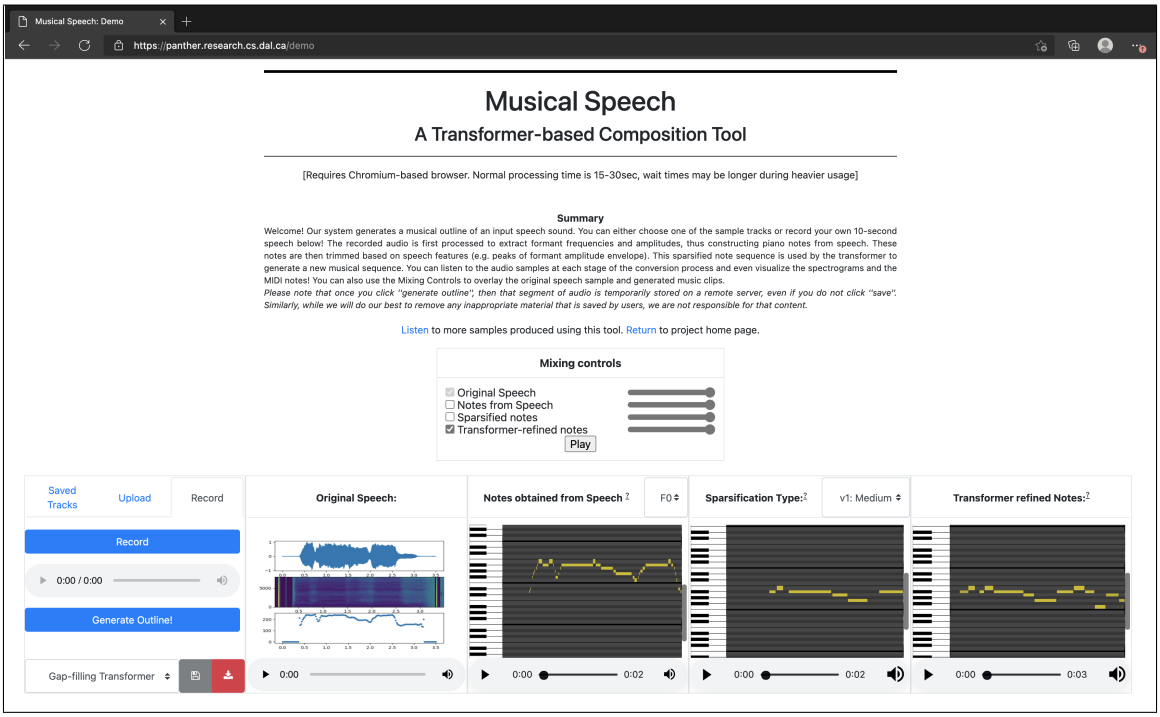

Figure 4: The web interface for interacting with the musical speech system.

Figure 4 shows the user interface of the online interactive version of the system, which enables the following functionality:

1. The UI allows for the composers to either record their live speech using a microphone or to upload a pre-recorded audio file.
2. The chosen audio file will be processed as described above, with all the intermediate results made available in the MIDI format as shown.
3. Depending on the speech sample and the kind of inspiration that the composers are looking for, certain configurations may be more preferred; also keeping in mind that it is not easy to create a one-size-fits-all pipeline, we allow the users to choose the type of model, the formant frequencies to extract, and the sparsification method to choose.
4. The composers can interactively mix input speech, the intermediate MIDI results, and the generated final MIDI sample while the mixed output plays on a loop until they identify the desired proportions.
5. In order to use the generated samples in their compositions, the composers could choose to either download the MIDI samples corresponding to a particular configuration or all possible configurations.

8. Evaluation

Creativity support tools have been referred to as a "Grand Challenge for HCI Researchers" (Shneiderman, 2009). A significant part of this challenge, especially as the tools themselves are becoming more powerful, is also in their evaluation. Recently, Remy et al. (2020) surveyed over 100 papers presenting creativity support tools, with the focus of how to evaluate such tools. They distinguish, for example, between evaluating usability and creativity: "traditional usability methods emphasize aspects such as efficiency, precision, error prevention, and adherence to standards [..], but do not address core dynamics of creative work, such as exploration, experimentation, and deliberate transgression of standards". Arguably, the latter considerations are indeed far more relevant in the present work than the former ones. While Remy et al indicate that the task of evaluating creativity support tools is very much an open problem, they do propose a set of recommendations on how to evaluate them in future. These recommendations include recruiting domain experts (in this case, professional musicians) and considering longitudinal, in-situ studies. (41% of the evaluations they surveyed lasted less than or up to one hour, and altogether 82% lasted less than or up to a day; neither of these would generally qualify as longitudinal).

Recognizing, then, that a rigorous such evaluation is outside the scope of this paper, we present two related analyses. First, in Section 8.1, a quantitative analysis briefly examines the question of whether the musical language modeling aspect of our tool is doing what it purports to be doing, e.g. does it shift the statistics of speech-based sequences towards those of musical sequences?

Second, in Section 8.2, we present a few samples created by professional musicians, one of whom spent many hours across several weeks experimenting with the system. Many more samples were created during that time; we choose just a small and diverse set here. We also note that some of the design choices made were in fact done so collaboratively with the expert musician, by providing them with early versions of the tool and incorporating their feedback (a well-known approach for effective design processes (Rogers et al., 2011)).

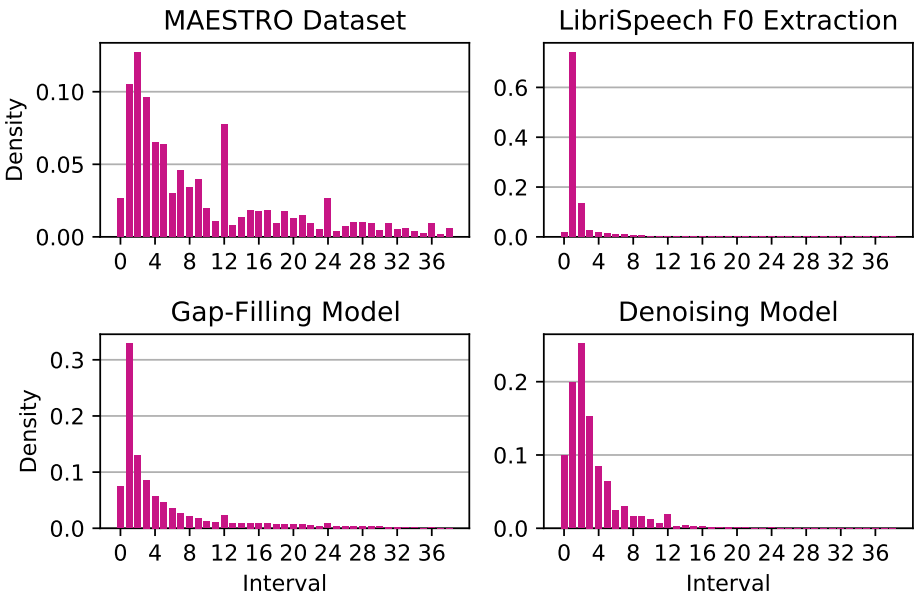

Figure 5: Frequency of musical intervals in the monophonic MAESTRO dataset, the F_0 values extracted from LibriSpeech samples, and the outputs of the gap-filling and denoising model respectively.

8.1. Quantitative Analysis

To evaluate the effectiveness of our models, we compared statistical features of the MAESTRO dataset against the Transformer outputs. In these experiments, we used the test-clean partition of the LibriSpeech dataset (Panayotov et al., 2015) as a source of voice recordings. In particular, we looked at the distribution of the intervals between adjacent pitches in sequences. Figure 5 shows the intervallic distributions for the skyline of the MAESTRO dataset, the extracted F_0 sequences, the outputs of the gap-filling model with medium-level syllable-nuclei sparsification applied, and the outputs of the denoising model.

The distribution of intervals from the MAESTRO dataset are indicative of common patterns in classical music. For example, small intervals of 1, 2, or 3 semitones are the most common kind of musical movement. An interval of 6 semitones is uncommon relative to 5 or 7 semitones (in music terms, a tritone, a perfect 4th, and a perfect 5th respectively), whereas 12 semitones is common relative to 11 or 13 semitones (an octave, a major 7th, and a minor 9th respectively). The shape of the distribution is roughly periodic with a decreasing scale, with a period of 12 semitones.

Extracted F_0 values from speech are highly chromatic (intervals of 1 semitone are frequent) and this is reflected in the distribution for LibriSpeech F_0 speech sequences. However, the distributions for the model outputs lie somewhere in the middle of the spectrum between pure music data and extracted F_0 values. The pipeline utilizing the gap-filling model maintains portions of the chromatic F_0 sequence, but noticeably increases the occurrence

of larger interval sizes. The denoising model is not constrained to keep any part of the F_0 sequences, resulting in a distribution closer to that of the MAESTRO sequences.

8.2. How is this used by musicians?

To examine how this system is used by musicians, we provide a set examples of different musical excerpts created with the system illustrating a set of use cases. Note that for some of these examples, listening with headphones is strongly recommended.

8.2.1. Use Case 1 - A Fun Tool for Direct Speech to Melody

Fun to have fun ([link]): This example demonstrates a musician using the system to sketch out an accompaniment for a speech recording of a passage from a children's book Geisel et al. (1957). First we hear the recording on its own, followed by the dense sequence of F_0 values (0:07). The denoising Transformer rewrote the extracted frequencies into a new melody (0:14). The musician added a simple rhythm section (0:21), and adjusted the melody slightly, still roughly aligned harmonically and rhythmically with the original voice (0:27). The result is a melody, harmonic, and rhythmic accompaniment that very closely follows and "supports" the contour of the original spoken voice recording.

8.2.2. Use Case 2 - Direct Speech to Sound

Hi. I'm a robot. I love you. ([Part 1 Link]): In this example, a musician uses all of the formants to create sounds that closely match the sounds of the original words. The format of this audio file is 3 subsequences, each subsequence itself consisting of 3 parts: (speech) (formantV1) (formantV2). That is, (formantV1) uses a musical synthesizer to render the full set of formants corresponding to (speech). (formantV2) is exactly the same, but rendered using a different musical synthesizer. The clip consists of 3 such subsequences. This gives us a sense of the range of sounds that can be produced using a single piece of text, and the impact of choice of instrument used.

Hi. I'm a robot. I love you. ([Part 2 Link]): In this one, the same speech was used, and all of the musical material was generated by the gap-filling (V2) music-to-speech system using the F0 output, and then orchestrated and mixed by a musician so that it "worked" as a short musical clip. It was noted that while the system generated the material quickly, it was still a slow, careful process for the musician to work with that raw material to bring it to this point.

8.2.3. Use Case 3 - Musical Material Generator

Moo-cow (link): A musician created this piece by incorporating the MIDI files our system generated from a ten-second spoken voice recording (itself also included throughout the piece). The singing is the only audio recording added other than the original voice and the generated MIDI files. That is, *all raw MIDI files* (i.e. all non-vocal clips used for instruments and synths) in this piece were generated by our system. They were then assembled, orchestrated, and mixed to create and produce this resulting 30-second excerpt.

To provide insight into the musical process, Appendix A shows some parts of the score of the track as it appeared on the musician's digital workstation software.

Thing 1 and Thing 2 (link): This is another example similar to the previous, where a musician has arranged and orchestrated the original speech recording and MIDI files to create the track, with all raw MIDI files generated by the system.

9. Conclusion

We have proposed a pipeline for translating speech into musical building blocks as an interactive tool for musical composition and illustrated its effectiveness through examples of music created with the help of our tool. In achieving this, we make novel use of speech-processing techniques to effectively sidestep the need for a paired speech-music dataset.

Our evaluation reveals that different training objectives and architectural choices can give rise to different forms of dependence on the conditioning input. We use a set of created musical excerpts to demonstrate aspects of the system and examples of how it can be effectively used to generate new musical pieces.

10. Acknowledgements

We thank Rudolf Uher and group for insightful discussions. We thank the Canadian Institute for Advanced Research (CIFAR) for their support. Resources used in preparing this research were provided, in part, by NSERC, the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute www.vectorinstitute.ai/#partners. We thank the Magenta team at Google for making the MAESTRO dataset publicly available.

References

Ableton AG. Ableton Live 10. Software, 2019. URL https://www.ableton.com/en/products/live-lite/.

Ulli Beier. The talking drums of the yoruba. *African Music: Journal of the African Music Society*, 1, 1954.

Paul Boersma. Praat, a system for doing phonetics by computer. *Glot. Int.*, 5(9):341–345, 2001.

Richard Boukas. Hermeto Pascoal: Visionary of Contemporary Brazilian Music. Lecture, 2013. URL https://www.youtube.com/watch?v=7RUlbU6CMx0.

Mason Bretan, Sageev Oore, Jesse Engel, Douglas Eck, and Larry Heck. Deep music: Towards musical dialogue. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017. URL https://ojs.aaai.org/index.php/AAAI/article/view/10544.

Pablo Samuel Castro. Performing structured improvisations with pre-trained deep learning models. *arXiv preprint arXiv:1904.13285*, 2019.

Xiaotong Cheng, Xiaoxia Wang, Tante Ouyang, and Zhengzhi Feng. Advances in emotion recognition: Link to depressive disorder. In *Mental Disorders*. IntechOpen, 2020.

Alain De Cheveigné and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

Nivja H De Jong and Ton Wempe. Praat script to detect syllable nuclei and measure speech rate automatically. *Behavior research methods*, 41(2):385–390, 2009.

Li Deng, Leo J Lee, Hagai Attias, and Alex Acero. Adaptive kalman filtering and smoothing for tracking vocal tract resonances using a continuous-valued hidden dynamic model. *IEEE transactions on audio, speech, and language processing*, 15(1):13–23, 2006.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

Sander Dieleman, Aaron van den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In *Advances in Neural Information Processing Systems*, pages 7989–7999, 2018.

Nai Ding, Aniruddh D Patel, Lin Chen, Henry Butler, Cheng Luo, and David Poeppel. Temporal modulations in speech and music. *Neuroscience & Biobehavioral Reviews*, 81: 181–187, 2017.

Chris Donahue, Ian Simon, and Sander Dieleman. Piano genie. In *24th International ACM Conference on Intelligent User Interfaces (ACM IUI 2019)*, 2019.

Sri Harsha Dumpala, Jason d'Eon, and Sageev Oore. Sine-wave speech as pre-processing for downstream tasks. In *International Symposium on Frontiers of Research in Speech and Music*, 2020.

Sri Harsha Dumpala, Sheri Rempel, Katerina Dikaios, Mehri Sajjadian, Rudolf Uher, and Sageev Oore. Estimating severity of depression from acoustic features and embeddings of natural speech. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

Jean-Louis Durrieu and Jean-Philippe Thiran. Source/filter factorial hidden markov model, with application to pitch and formant tracking. *IEEE transactions on audio, speech, and language processing*, 21(12):2541–2553, 2013.

Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. DDSP: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*, 2020.

J Epps, J R Smith, and J Wolfe. A novel instrument to measure acoustic resonances of the vocal tract during phonation. *Measurement Science and Technology*, 8(10):1112–1121, oct 1997. doi: 10.1088/0957-0233/8/10/012.

Theodor Seuss Geisel et al. *The cat in the hat*. Random House Books for Young Readers, 1957.

Maija Hausen, Ritva Torppa, Viljami R Salmela, Martti Vainio, and Teppo Särkämö. Music and speech prosody: a common rhythm. *Frontiers in psychology*, 4:566, 2013.

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. In *International Conference on Learning Representations*, 2019.

Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. In *International Society for Music Information Retrieval (ISMIR)*, 2017.

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: Generating music with long-term structure. In *International Conference on Learning Representations*, 2018a.

Cheng-Zhi Anna Huang, Hendrik Vincent Koops, Ed Newton-Rex, Monica Dinculescu, and Carrie J Cai. Ai song contest: Human-ai co-creation in songwriting. *arXiv preprint arXiv:2010.05388*, 2020.

Sicong Huang, Qiyang Li, Cem Anil, Xuchan Bao, Sageev Oore, and Roger B Grosse. Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer. *arXiv preprint arXiv:1811.09620*, 2018b.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 8:64–77, 2020. URL https://transacl.org/ojs/index.php/tacl/article/view/1853.

Peter Ladefoged and Keith Johnson. *A course in phonetics*. Nelson Education, 2014.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.703. URL https://doi.org/10.18653/v1/2020.acl-main.703.

Ryan Louie, Andy Coenen, Cheng Zhi Huang, Michael Terry, and Carrie J Cai. Novice-ai music co-creation via ai-steering tools for deep generative models. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

John Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4): 561–580, 1975.

Nicholas Meade, Nicholas Barreyre, Scott C Lowe, and Sageev Oore. Exploring conditioning for generative music systems with human-interpretable controls. In *International Conference on Computational Creativity*, 2019.

Paul Mermelstein. Automatic segmentation of speech into syllabic units. *The Journal of the Acoustical Society of America*, 58(4):880–883, 1975.

Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. *arXiv preprint arXiv:1805.07848*, 2018.

Masanori Morise, Hideki Kawahara, and Haruhiro Katayose. Fast and reliable f0 estimation method based on the period extraction of vocal fold vibration of singing voice and speech. In *Audio Engineering Society Conference: 35th International Conference: Audio for Games*. Audio Engineering Society, 2009.

Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. World: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE TRANSACTIONS on Information and Systems*, 99(7):1877–1884, 2016.

Daniel Oore. Trump: The Musical Prophet. In Eric T Kasper and Benjamin S Schoening, editors, *The Role of Music in the 2016 presidential election and beyond*. Denton: University of North Texas Press, 2018.

Daniel Oore. Trump: the musical prophet, 2021. URL https://www.youtube.com/playlist?list=PL0-rkS6BcMVw2VW3-4WITmUNw-mSeji3J.

Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, 32(4):955–967, 2018.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

Aniruddh D Patel. *Music, language, and the brain*. Oxford university press, 2010.

Christine Payne. Musenet. OpenAI Blog, 2019.

Thilo Pfau and Günther Ruske. Estimating the speaking rate by vowel detection. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 2, pages 945–948. IEEE, 1998.

Hartmut R Pfitzinger. Local speech rate perception in german speech. In *Proc. of the XIVth Int. Congress of Phonetic Sciences*, volume 2, pages 893–896, 1999.

Robert F Port. Meter and speech. *Journal of phonetics*, 31(3-4):599–611, 2003.

L Rabiner and BH Juang. Fundamentals of speech recognition. *Englewood Cliffs Publisher, New Jersey*, pages 200–232, 1993.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Steve Reich. Different trains. Musical Composition for String Quartet and Tape, 1988.

Christian Remy, Lindsay MacDonald Vermeulen, Jonas Frich, Michael Mose Biskjaer, and Peter Dalsgaard. Evaluating creativity support tools in hci research. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, DIS '20, page 457–476, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369749.

Adam Roberts, Jesse Engel, Curtis Hawthorne, Ian Simon, Elliot Waite, Sageev Oore, Natasha Jaques, Cinjon Resnick, and Douglas Eck. Interactive musical improvisation with magenta. In *Proc. NIPS*, 2016.

Adam Roberts, Jesse H Engel, Sageev Oore, and Douglas Eck. Learning latent representations of music to generate interactive musical palettes. In *Intelligent User Interfaces (IUI) Workshop on Intelligent Music Interfaces*, 2018.

Yvonne Rogers, Helen Sharp, and Jenny Preece. *Interaction design: beyond human-computer interaction*. John Wiley & Sons, 2011.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. Leveraging pre-trained checkpoints for sequence generation tasks. *Trans. Assoc. Comput. Linguistics*, 8:264–280, 2020. URL https://transacl.org/ojs/index.php/tacl/article/view/1849.

Ben Shneiderman. *Creativity Support Tools: A Grand Challenge for HCI Researchers*, pages 1–9. Springer London, London, 2009.

Charles Spearin and Band. The happiness project, 2009.

Aiva Technologies. Aiva - the artificial intelligence composing emotional soundtrack music. Website, 2020. URL https://www.aiva.ai/.

Sean Vasquez and Mike Lewis. Melnet: A generative model for audio in the frequency domain. *arXiv preprint arXiv:1906.01083*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence

d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html.

B Yegnanarayana and K Sri Rama Murty. Event-based instantaneous fundamental frequency estimation from speech signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):614–624, 2009.

Yaodong Zhang and James R Glass. Speech rhythm guided syllable nuclei detection. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3797–3800. IEEE, 2009.

Figure 6: MIDI files labeled "overwrite-gap-fill" (green), trigger a moon guitar [aka yueqin] (labeled "MOONTAR")

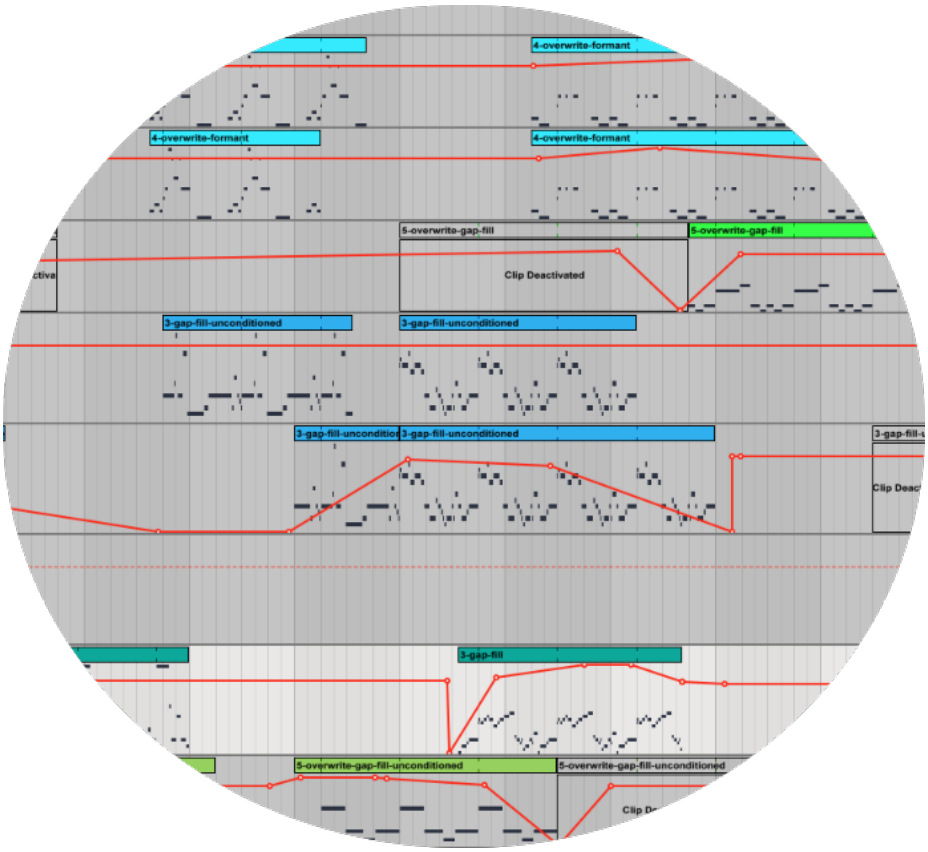

Figure 7: The MIDI files (green– and blue-tabbed files containing tiny black rectangles of different heights and lengths representing pitch and duration information respectively, and which trigger the MIDI instruments in the tracks that they populate) are each labeled along their green and blue tabs according to the manner in which the given raw MIDI file was generated.

Appendix A. Sample Elements of a Composition

Figures 6 and 7 illustrate some elements of the *Moocow* composition described in Section 8.2.3.

Appendix B. Syllable-nuclei detection

A brief description of the steps to locate syllable nuclei positions in speech is as follows:

1. Extract the loudness contour from the speech signal. The extracted loudness contour is Smoothed using moving-average smoothening technique.

2. All peaks in the smoothed loudness contour are considered as the initial set of potential syllable nuclei.

3. Discard all peaks below a pre-fixed threshold. Here we set this threshold to be 2 dB above the median loudness measured over the entire input speech recording.

4. Inspect the preceding dip in intensity of loudness contour for each peak. Consider only peaks with a preceding dip of at least 2dB with respect to the current peak as a potential syllable nuclei and discard other peaks.

5. Extract the F_0 contour to obtain the voiced/unvoiced regions in the speech signal. Regions with F_0 values below 40 are considered as unvoiced regions. Discard peaks within the unvoiced regions.

6. Peaks retained after step-4 are considered as syllable nuclei.