

# STEMGEN: A MUSIC GENERATION MODEL THAT LISTENS

Julian D. Parker    Janne Spijkervet\*    Katerina Kosta\*    Furkan Yesiler  
 Boris Kuznetsov    Ju-Chiang Wang    Matt Avent    Jitong Chen    Duc Le

SAMI, ByteDance Inc.

## ABSTRACT

End-to-end generation of musical audio using deep learning techniques has seen an explosion of activity recently. However, most models concentrate on generating fully mixed music in response to abstract conditioning information. In this work, we present an alternative paradigm for producing music generation models that can listen and respond to musical context. We describe how such a model can be constructed using a non-autoregressive, transformer-based model architecture and present a number of novel architectural and sampling improvements. We train the described architecture on both an open-source and a proprietary dataset. We evaluate the produced models using standard quality metrics and a new approach based on music information retrieval descriptors. The resulting model reaches the audio quality of state-of-the-art text-conditioned models, as well as exhibiting strong musical coherence with its context.

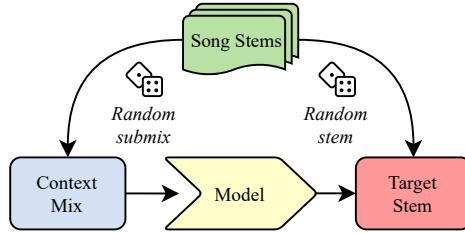
**Index Terms**— Music Generation, Deep Learning, LLMs, Generative Models

## 1. INTRODUCTION

End-to-end generation of musical audio using deep learning techniques is a fairly new discipline, the formative work having arguably started with WaveNet [1]. Recent work has shown a large jump in the quality and diversity of generated music by borrowing techniques from the image and language processing fields. Some approaches operate on tokenized audio representations [2, 3] using techniques from the large language model (LLM) literature [4, 5, 6]. Other approaches use score-matching techniques to generate audio directly [7, 8], or encoded as a continuous latent representation [9].

Music can generally be thought of as the sum of a number of independent but strongly related individual parts, conceived at different levels of elaboration and textural layout which are combined together to produce a full piece of music [10, p. 194]. A part may correspond to a musician playing a particular instrument, or to something more abstract like the output of a sampler or synthesizer. These parts are often colloquially referred to as *stems*. Music production can be thought of as an iterative procedure that seeks to add and refine these stems to fit the aesthetic preferences of the producer or musician. In order for the resulting mixed music to sound coherent, each of these stems must be constructed to be sympathetic with the context of the existing composition, both musically and texturally. A generative model that performs this task would be a desirable tool for those making music, and one which could support existing creative workflows instead of supplanting them.

Most existing generative models for musical audio are conditioned on abstract information, varying from text descriptions [4, 5, 7] to style categories [11], and cannot be used easily for this iterative composition approach. Models that can listen to the context audio directly and generate a musically appropriate response are rarer but



**Fig. 1:** Schematic diagram of the StemGen training paradigm.

have been proposed in the context of generating musical accompaniments for singing [12] and in the context of generating multi-track instrumental music [13]. However, these models place fixed semantic restrictions on both the context audio and the generated response.

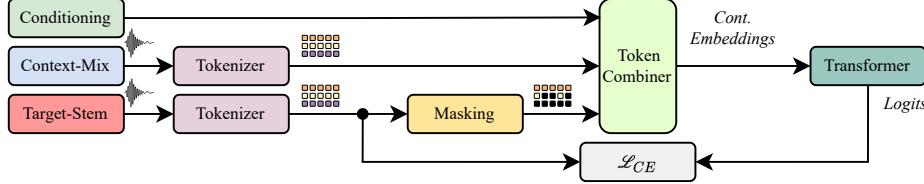
In this work we propose a novel training paradigm for producing end-to-end generative models that are able to listen to a musical context and generate an appropriate response. We propose a network architecture for implementing such a model, based on a non-autoregressive language model with a transformer backbone. This architecture is extended from previous literature [6] with a novel approach to combining multiple streams of audio tokens. We introduce two novel improvements to generation: *multi-source classifier-free-guidance*, and *causal-bias* during iterative decoding. We train the architecture on two datasets of stem-based musical audio and evaluate the resulting models using standard objective metrics, a novel approach using an amalgamation of music information retrieval (MIR) metrics, and listening tests.

## 2. MODELLING APPROACH

Given a dataset that contains pieces of music separated into their stems, we can construct pairs of data consisting of a *context-mix*, which represents the musical context, and a *target-stem*, which represents a response to that context. If the piece of music contains  $M$  stems, the context-mix can be constructed as a mixture of  $N < M$  stems selected at random from the full set. The target-stem can then be taken to be a single stem, selected at random from those not included in the context-mix. This procedure means that a single song with  $N$  stems can result in  $\frac{N}{2}(2^N - 2)$  training pairs. This acts as a intrinsic form of data augmentation, and likely somewhat mitigates the scarcity of high quality music data in stem separated format.

The training problem can be formulated in one of two ways, either as learning the joint distribution  $p(t, a)$  of the context-mixes,  $a$ , and target-stems,  $t$ , or as learning the conditional distribution of target-stems given the context-mixes,  $p(t|a)$ . We refer to modelling  $p(t|a)$  as *conditional generation* and modelling  $p(t, a)$  as *joint generation*. In this work, we address the conditional generation problem. Fig. 1 shows an overview of this paradigm.

\*Equal contribution



**Fig. 2:** Schematic showing overall architecture of the StemGen model during training.

### 2.1. Model Architecture

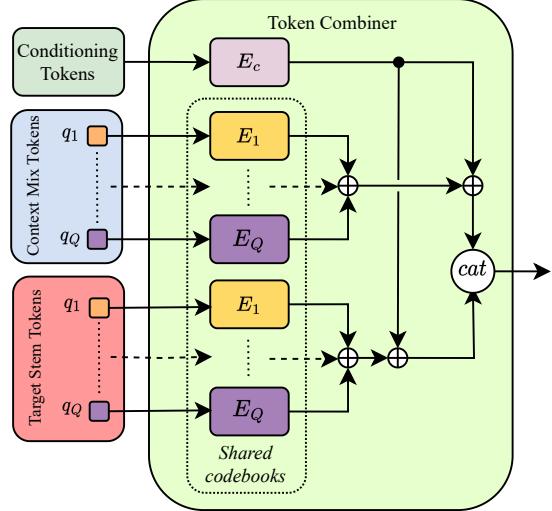
As with many recent models [4, 5, 6], we reformulate the problem as modelling sequences of abstract tokens rather than modelling sampled waveforms. This allows the application of powerful techniques from the domain of language modelling. In order to produce such a sequence of tokens from audio waveforms, we use a separate audio encoding model [2]. This model employs a residual vector quantizer (RVQ), which means that each temporal frame of the signal is described by a set of multiple hierarchical tokens. There exist different strategies for modelling this token sequence. MusicLM [4] separates the tokens into coarse and fine sets, flattens each set into a single stream with one token per-step, and then models them autoregressively. MusicGen [5] also takes an autoregressive approach, but presents a number of methods for combining all the tokens for a time-step into a single sequence element. VampNet [6], which is derived from the speech model SoundStorm [14], also combines some tokens into a single sequence element (albeit split into coarse and fine groups in a manner similar to MusicLM), but then models the resulting two sequences in a non-autoregressive manner, similar to a masked language model. Our approach is similar to MusicGen in that it employs a single model instead of separate coarse/fine models, and combines all token levels into a single sequence element. However we employ a non-autoregressive approach to training and sampling, which is more similar to VampNet. The overall structure of the architecture during training is shown in Fig. 2

In contrast to VampNet & MusicGen, we must combine multiple channels of audio (context-mix and target-stem) into a single sequence element. We achieve this by producing two sets of embeddings, one for each audio channel. Each embedding is produced by summing the embeddings for each hierarchical token level for that audio channel. These two embeddings are then concatenated along the embedding dimension, and used as a single sequence element. Before concatenation, embeddings are also calculated for any relevant non-audio conditioning. This could consist of encoded text embeddings, embeddings from a model such as CLAP [15], or embeddings produced from available categorical metadata such as instrument type or musical genre. This conditional embedding is then summed into the embeddings for each audio channel, before concatenation. Fig. 3 illustrates this token combination process.

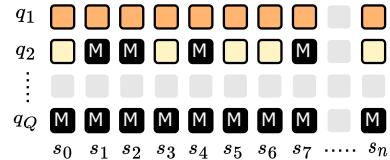
Training is conducted using the masking procedure of SoundStorm [14], but with one modification – we ensure that when training a particular hierarchical token level, all lower hierarchical levels (corresponding to finer residuals in the RVQ) are completely masked. Fig. 4 illustrates this. This masking procedure is only applied to the tokens for the target-stem. Given that we want to train a *conditional-generation* model, context-mix tokens are left completely unmasked. If we wanted to train a *joint-generation* model, the context-mix tokens could be included in the masking procedure.

#### 2.1.1. Causally biased iterative decoding

To sample new outputs using a trained masked-LM, an iterative process is used. For a particular token level, we start with the entire sequence masked. Candidate samples are then generated for the en-

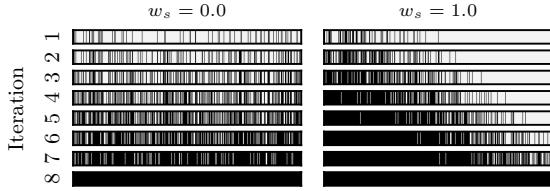


**Fig. 3:** Schematic showing how the  $Q$  RVQ levels of both the context-mix and the target stem are converted into continuous embeddings using codebooks  $E_1 \dots Q$ , and then combined with each other along with conditioning information.



**Fig. 4:** Example masking pattern when training to generate the 2nd level of a  $Q$ -level audio tokenizer.  $q_1 \dots Q$  denote the RVQ levels, and  $s_0 \dots n$  the sequence elements (equivalent to time steps in this case).

tire sequence by sampling from the logits predicted by the backbone of the masked-LM for the current state of the sequence. These candidate samples are then ranked according to some criteria (see below), and the top-k are retained. The process is then repeated using the updated sequence, until all tokens are unmasked. It is then repeated for the next hierarchical token level. The criteria used to rank the candidate samples at each step has a strong effect on the quality of generated output. SoundStorm ranks samples by the confidence of the model, as given by the value of the logit for the sampled token. VampNet extends this by adding Gumbel noise to the confidence rankings with a defined weighting. We found that these sampling procedures produced poor output in many cases. Biassing the generation towards confidence leads to monotonous and uninteresting output, whereas relying too heavily on random selection leads to poor transients and unnatural amplitude fluttering in the output. We propose an extension to this sampling approach that encourages earlier sequence elements to be sampled first, enforcing a kind of fuzzy



**Fig. 5:** Example of iterative decoding for sequence of a single token level over 8 iterations, with and without causal-bias. Black denotes which sequence elements have been sampled.

causality. The resulting ranking function is given by:

$$\rho(x_n) = w_c.c(x_n) + w_s.(1 - n/N) + w_r X \quad (1)$$

where  $x_n$  is the candidate sample at sequence index  $n$ ,  $N$  is the total sequence length,  $c(x_n)$  is the model's confidence in the candidate sample as calculated by applying softmax to the logits,  $X \sim U(0, 1)$  is a uniformly distributed random variable, and  $w_p, w_s, w_r$  are scalar weights. In Fig. 5 we show the evolution of a single sequence during iterative decoding, with and without causal-bias.

### 2.1.2. Multi-source classifier-free-guidance

We employ classifier-free-guidance [16] during the sampling of candidate tokens. In addition to applying this process to the non-audio conditioning, we also apply it to the audio context conditioning, with the hope of enforcing stronger alignment to the audio context. The classifier-free-guidance can be applied over both conditioning sources simultaneously using the standard formulation:

$$\log(p(\mathbf{t})p(\mathbf{c}|\mathbf{t})^\lambda) \approx \lambda \log(p(\mathbf{t}|\mathbf{c})) + (1 - \lambda) \log(p(\mathbf{t})) \quad (2)$$

where  $\mathbf{c}$  is a combined conditioning source containing both the context-mix and any other conditioning and  $\lambda$  is the guidance scale.

We also introduce a technique for weighting the guidance for the multiple conditioning sources independently:

$$\begin{aligned} \log(p(\mathbf{t}) \prod_{i=1}^N p(\mathbf{c}_i|\mathbf{t})^{\lambda_i}) &\approx \sum_{i=1}^N \lambda_i \log(p(\mathbf{t}|\mathbf{c}_i) \\ &+ (1 - \sum_{i=1}^N \lambda_i) \log(p(\mathbf{t})) \end{aligned} \quad (3)$$

where the  $\mathbf{c}_i$  are independent conditioning sources, and  $\lambda_i$  are the guidance scales for each conditioning source. The derivation follows trivially from that of single-source classifier-free guidance, via the application of Bayes' rule. Note that it is important to train the model with independent dropout for each conditioning source.

## 3. EXPERIMENTS

All models are trained on a single NVIDIA A100 GPU with 80GB of VRAM. The batch size is 40. Training proceeds until validation accuracy has clearly stopped increasing for 20k steps, which was generally after around 150k total training steps. This translates to around 4 days of training time. The AdamW optimizer is used, with a tri-stage learning rate schedule consisting of 10k warmup steps, 50k steps at the maximum learning rate of 0.0005, and then a slow exponential decay over 300k steps.

As a baseline, we train on the Slakh [17] dataset, which consists of 145 hours of synthetic musical audio separated into stems. In addition, we train on an internal dataset of 500 hours of licensed

human-played music separated into stems. Individual songs were chunked into 1 minute segments during data preparation. When constructing training pairs, a random 20s crop of each segment is chosen and the random submixing procedure described above in Sec. 2 is applied. The resulting context-mix and target-stem are checked to make sure they are not silent, and discarded if so. Both datasets contain metadata specifying the instrument type of each stem, with classes corresponding to the 18 General MIDI instrument categories [18]. We use a trainable codebook with 18 entries to translate this conditioning information into an embedding. This model therefore has two conditioning sources: the context-mix and the instrument category, referred to as  $\mathbf{c}_a$  and  $\mathbf{c}_i$  respectively.

We use the publicly available checkpoint of Encodec [2] also used for MusicGen [5]. This codec operates on 32kHz audio, and datasets are resampled appropriately. The encoded audio has a frame rate of 50Hz, with four tokens in the range 0...2047 per frame.

As with any LM-like structure, the above described model architecture is broadly agnostic to the network type used for the main decoder block. In practice, most models of this type employ a transformer. We follow this convention and employ a LLaMa-type transformer [19] for the experiments presented here. The transformer has an embedding dimension of 1024 and the number of attention heads is 16. 20 layers are used, giving a total parameter count of  $\sim 250M$ .

## 3.1. Evaluation Metrics

We analyze the audio samples generated by our system in two objective evaluation strands. The first is Fréchet Audio Distance (FAD) [20], which we calculate on VGGish embeddings as per the original paper (referred to in results as *FAD*). The second is the Music Information Retrieval Descriptor Distance (MIRDD). Similarly to FAD, MIRDD is calculated on two populations. Instead of comparing distributions in an abstract embedding space, MIRDD compares the distributions with respect to a number of MIR (pitch-, rhythm-, and structural-based) descriptors.

Most of the descriptors are inspired by the research for melody expectation and prediction. The use of such musical quantities has been shown to enhance melodic expectation when embodied in a cognitively plausible system for music prediction and demonstrated as a successful system for evaluating objectively music generation models that fit subjective input (for more details see [21, 22]). Descriptors have been developed for evaluating monophonic generations from composition models in MIDI format [23]. We expand the concept by introducing a set designed to evaluate generated audio in both monophonic and polyphonic manner.

We compute the descriptors for every audio in the reference and test populations, utilising the outcome of the models for audio transcription [24], beat and downbeat detection [25], key and chords estimation [26], as well as structure extraction [27]. This gives us a set of vectors of real values (for continuous quantities) or integer counts (for discrete quantities). MIRDD is computed, extracting the KL-Divergence of the probability distribution pairs per descriptor and averaging the values. A lower MIRDD score is considered better. The list of descriptors used is the following: key signature, length of note pitch range, amount of unique pitch classes, maximum vertical density of percussive notes (computed in a 0.6s window size), number of beats per bar, chord labels variety, chord tonality alteration ratio (moving from minor to major and vice versa), note pitches, note pitch classes, chord triad labels and structure labels.

## 3.2. Results

For evaluation, we generated sets of 400 example outputs from each model. The examples were produced following a similar procedure

to that used for the construction of training examples, but pulling from a separate set of test data not contained within the training set. For each example we generate a context-mix, and randomly choose a target-stem category. The generated stem produced by the model for this set of conditioning is placed into one population. The real stem for the matching target category is placed into another population. These two populations can then be compared with a variety of metrics. This procedure ensures there is no systematic error introduced by an imbalance in target categories between the two populations. We do not perform pairwise evaluation, but instead compare the distributions of the two populations. FAD-type metrics are calculated between the two populations of isolated stems, whereas MIRDD is calculated on modified populations consisting of stems summed with their original context-mix. This allows the MIRDD metric to better penalize poor musical alignment and coherence.

### 3.2.1. Sampling hyper-parameters & ablations

In order to validate our novel improvements to the decoding procedure, we perform two ablations. For these ablations we use the model trained on the Slakh dataset. Firstly, we test the impact of multi-source classifier-free-guidance, as described in Sec. 2.1.2, by calculating metrics over a variety of guidance scales,  $\lambda_a$  and  $\lambda_i$ , with respect to our two conditioning sources  $\mathbf{c}_a$  and  $\mathbf{c}_i$ . The results can be seen in Tab. 1. A value of 1.0 for guidance scale is equivalent to no guidance with respect to that conditioning source. The exact best values for guidance scale are very dependent on the conditioning information, so for cases where the guidance scale is greater than 1.0 we show results averaged over 1000 examples at different values of  $\lambda_i$ ,  $\lambda_a$  up to a maximum of 4.0. The results confirm that adding guidance over multiple sources is beneficial. We settle on  $\lambda_a = \lambda_i = 3.0$  as a general setting for evaluation.

	FAD	MIRDD
$\lambda_i = \lambda_a = 1.0$	4.30	0.41
$\lambda_i > 1.0, \lambda_a = 1.0$	3.66	0.26
$\lambda_i = 1.0, \lambda_a > 1.0$	3.20	0.29
$\lambda_i > 1.0, \lambda_a > 1.0$	<b>3.17</b>	<b>0.25</b>

**Table 1:** Averaged objective metrics with different ranges of guidance scales.

Secondly, we test the impact of causal bias during iterative decoding by comparing various relative strengths of causal-bias. The other ranking weights,  $w_c$  and  $w_r$  are set to 0.1 and 1.0 respectively. A causal-bias weight  $w_s$  of 0 gives a decoding scheme similar to that used in VampNet [6]. The results can be seen in Tab. 2. We can see that adding a small amount of causal-bias has a positive effect on FAD and MIRDD metrics, indicating an increase in sound quality and musical alignment. This lessens as  $w_s$  increases further. We therefore settle on  $w_s = 0.1$  for further experiments.

$w_s$	0.0	0.1	0.2	0.5
FAD	3.18	<b>3.12</b>	3.13	3.17
MIRDD	0.32	<b>0.14</b>	0.18	0.20

**Table 2:** Objective evaluation metrics for various values of causal-bias weight  $w_s$ .

During iterative decoding, we use 128, 64, 32 and 32 steps respectively for the four hierarchical levels of the tokenizer. This is universal across all results presented here.

### 3.2.2. Final evaluation

We evaluate both the model trained on Slakh, and that trained on our internal dataset. Both models are sampled from using a *best* set of sampling parameters derived above in Sec. 3.2.1. Additionally we sampled from each model using a set of *naive* sampling parameters, which are equivalent to removing classifier-free-guidance and causal-bias in decoding. We show objective metrics for these sets of outputs in Tab. 3. Whilst a direct comparison is not possible due to the different task, the FAD scores for the models trained on both Slakh and the internal dataset are consistent with those seen for state-of-the-art text-conditioned models [5]. It is also clear that the larger size and human-content of the internal dataset leads to an appreciable improvement in output quality. Examples from these models can be heard at the website associated with this work <sup>\*</sup>

	FAD	MIRDD
Slakh ( <i>naive</i> )	4.3	0.43
Slakh ( <i>best</i> )	3.12	0.14
Internal ( <i>naive</i> )	3.39	0.40
Internal ( <i>best</i> )	1.96	0.36

**Table 3:** Objective evaluation metrics for both models with *best* and *naive* sampling parameters.

We also conducted a listening test in the Mean Opinion Score (MOS) format, by asking 10 participants with music training to verify the subjective quality of the produced model. We constructed three sets of outputs by mixing generated or real stems with their corresponding context-mix. The generated stems were taken from the *naive* and *best* sets of output from the model trained on the internal dataset as evaluated in Tab. 3. The real stems were taken from the references sets used for previous evaluations. We collated a set of 60 mixes using this technique (equally split between *naive*, *best* and real) and asked listeners to rate the overall quality on a Likert scaled from very bad (1) to very good (5). The results are shown in Table 4, and confirm that the proposed model with *best* sampling parameters is capable of creating plausible musical outcomes.

	Real	Best	Naive
MOS	3.64	3.46	2.89

**Table 4:** MOS test results for real stems, and generated stems for the model trained on the internal dataset with *naive* and *best* sampling parameters.

## 4. CONCLUSIONS

In this work we defined a framework for training contextual music generation models using datasets of music split into stems. We presented a non-autoregressive language model-like architecture for implementing this, along with novel changes to support multiple audio channels, and novel sampling methods. We trained this architecture on two datasets, and presented ablations that verify the impact of our novel sampling changes. We evaluate the final models using objective metrics, a novel MIR-based metric, and with a subjective listening test. This evaluation shows that the audio quality of the resulting model is competitive with state-of-the-art music generation models, and that the musical alignment with context is good.

<sup>\*</sup><https://julian-parker.github.io/stemgen/>

## 5. REFERENCES

- [1] A. vd. Oord *et al.*, “WaveNet: A Generative Model for Raw Audio,” *arXiv*, 2016, 1609.03499.
- [2] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, “High Fidelity Neural Audio Compression,” *arXiv*, 2022, 2210.13438.
- [3] R. Kumar, P. Seetharaman, A. Luebs, I. Kumar, and K. Kumar, “High-Fidelity Audio Compression with Improved RVQ-GAN,” *arXiv*, 2023, 2306.06546.
- [4] A. Agostinelli *et al.*, “MusicLM: Generating Music From Text,” *arXiv*, 2023, 2301.11325.
- [5] J. Copet *et al.*, “Simple and Controllable Music Generation,” *arXiv*, 2023, 2306.05284.
- [6] H. F. Garcia, P. Seetharaman, R. Kumar, and B. Pardo, “VampNet: Music Generation via Masked Acoustic Token Modeling,” *arXiv*, 2023, 2307.04686.
- [7] Q. Huang *et al.*, “Noise2Music: Text-conditioned Music Generation with Diffusion Models,” *arXiv*, 2023, 2302.03917.
- [8] C. Hawthorne, I. Simon, A. Roberts, N. Zeghidour, J. Gardner, E. Manilow, and J. Engel, “Multi-instrument Music Synthesis with Spectrogram Diffusion,” *arXiv*, 2022, 2206.05408.
- [9] F. Schneider, Z. Jin, and B. Schölkopf, “Moūsai: Text-to-Music Generation with Long-Context Latent Diffusion,” *arXiv*, 2023, 2301.11757.
- [10] Nicholas Cook, *Music, Imagination, and Culture*, ACLS Humanities E-Book. Clarendon Press, 1990.
- [11] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A Generative Model for Music,” *arXiv*, 2020, 2005.00341.
- [12] C. Donahue *et al.*, “SingSong: Generating musical accompaniments from singing,” *arXiv*, 2023, 2301.12662.
- [13] G. Mariani, I. Tallini, E. Postolache, M. Mancusi, L. Cosmo, and E. Rodolà, “Multi-Source Diffusion Models for Simultaneous Music Generation and Separation,” *arXiv*, 2023, 2302.02257.
- [14] Z. Borsos, M. Sharifi, D. Vincent, E. Kharitonov, N. Zeghidour, and M. Tagliasacchi, “SoundStorm: Efficient Parallel Audio Generation,” *arXiv*, 2023, 2305.09636.
- [15] B. Elizalde, S. Deshmukh, M. Al Ismail, and H. Wang, “CLAP: Learning Audio Concepts From Natural Language Supervision,” *arXiv*, 2022, 2206.04769.
- [16] J. Ho and T. Salimans, “Classifier-Free Diffusion Guidance,” *arXiv*, 2022, 2207.12598.
- [17] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, “Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity,” in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019.
- [18] MIDI Manufacturers Association, “Complete MIDI 1.0 Detailed Specification,” <http://www.midi.org/techspecs/gm.php>, 1999/2008.
- [19] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, “LLaMA: Open and Efficient Foundation Language Models,” *arXiv*, 2023, 2302.13971.
- [20] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, “Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms,” in *Proc. Interspeech 2019*, 2019, pp. 2350–2354.
- [21] M. T. Pearce, *The construction and evaluation of statistical models of melodic structure in music perception and composition*, Ph.D. thesis, City University London, 2005.
- [22] M. Pearce and G. Wiggins, “Expectation in melody: The influence of context and learning,” *Music Perception*, vol. 23, pp. 377–405, 06 2006.
- [23] L.-C. Yang and A. Lerch, “On the evaluation of generative models in music,” *Neural Computing and Applications*, vol. 32, 05 2020.
- [24] W.-T. Lu, J.-C. Wang, and Y.-N. Hung, “Multitrack Music Transcription with a Time-Frequency Perceiver,” in *IEEE Int. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP)*, 2023, pp. 1–5.
- [25] Y.-N. Hung, J.-C. Wang, X. Song, W.-T. Lu, and M. Won, “Modeling Beats and Downbeats with a Time-Frequency Transformer,” in *IEEE Int. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP)*, 2022, pp. 401–405.
- [26] W.-T. Lu and J.-C. Wang and M. Won and K. Choi and X. Song, “SpecTNT: a Time-Frequency Transformer for Music Audio,” in *International Society for Music Information Retrieval Conference*, 2021.
- [27] J.-C. Wang, Y.-N. Hung, and J. B. L. Smith, “To Catch A Chorus, Verse, Intro, or Anything Else: Analyzing a Song with Structural Functions,” in *IEEE Int. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP)*, 2022, pp. 416–420.