EVALUATING GENERATIVE AI DRUMBEATS ON GUITAR TRACKS

-Kabir Gupta & Anoushka Yadav

PROBLEM STATEMENT

Creating music is often a collaborative process, but many guitarists—whether solo artists, independent musicians, or members of small bands—lack access to a dedicated drummer.

This limitation makes it difficult for them to compose full arrangements, practice effectively, or record high-quality tracks.

PROPOSAL & IMPACT

- A ML model that is capable of producing a drum track for an input guitar track.
- Democratizes music production by providing independent musicians and small bands access to high-quality drum tracks without needing a full band or professional studio.
- Saves time and reduces costs, allowing artists to focus on their compositions rather than technical barriers.
- Bridges the gap between creative vision and production reality, enabling artists to bring their musical ideas to life.

POTENTIAL APPLICATIONS

- Music Composition: Enable guitarists to create full-band arrangements quickly without needing advanced technical expertise or additional resources.
- Practice & Jamming: Generate dynamic drum tracks for practice sessions, offering a more immersive and productive experience.
- Recording & Production: Provide independent musicians and small studios with professional-quality drum tracks without the cost and complexity of hiring a live drummer.
- Live Performance: Act as a virtual drummer for solo artists during live shows, enhancing the performance with adaptive and high-quality drum accompaniment.

DRAWBACKS OF TRADITIONAL METHODS

1. Manually programming drum tracks in DAWs is time-consuming.
2. Pre-recorded loops lack adaptability and dynamic nuances.
3. Traditional methods often fail to capture the originality of compositions.
4. Hiring session drummers can be costly.
5. Professional drum recording services may be impractical for independent musicians.

LITERATURE REVIEW

CycleDRUMS: Automatic Drum Arrangement For Bass Lines Using CycleGAN

Giorgio Barnabò, Giovanni Trappolini, Lorenzo Lastilla, Cesare Campagnano, Angela Fan, Fabio Petroni, and Fabrizio Silvestri

CycleGAN struggles with capturing long-range dependencies

1. **CycleGAN operates on short 5-second mel-spectrograms, treating them as static image-like representations rather than sequential data.**
2. **Because it does not maintain memory across segments, it fails to capture overarching musical structures like transitions between verses, choruses, and fills.**

Giorgio Barnabò, Giovanni Trappolini, Lorenzo Lastilla, Cesare Campagnano, Angela Fan, Fabio Petroni, Fabrizio Silvestri. CycleDRUMS: Automatic Drum Arrangement For Bass Lines Using CycleGAN.

distribution of predicted class for these samples.

V. Conclusions and Future Work

In this work, we presented a novel approach to automatically produce drums starting from a bass line. We applied CycleGAN to real bass lines, treated as gray-scale images (mel-spectrograms), obtaining good ratings, especially if compared to another image-to-image translation approach (Pix2pix). Given the novelty of the problem, we proposed a reasonable procedure to properly evaluate our model outputs. Notwithstanding the promising results, some critical issues need to be addressed before a more compelling architecture can be developed. First and foremost, a larger and cleaner dataset of source separated songs should be created. In fact, manually separated tracks always contain a big deal of noise. Moreover, the model architecture should be further improved to focus on longer dependencies and to take into account the actual degradation of high frequencies. For example, our pipeline could be extended to include some recent work on quality-aware image-to-image translation networks [57], and spatial attention generative adversarial networks [58]. Finally, a certain degree of interaction and randomness should be inserted to make the model less deterministic and to give creators some control over the sample generation. Our contribution is nonetheless a first step toward more realistic and useful automatic music arrangement systems and we believe that further significant steps could be made to reach the final goal of human-level automatic music arrangement production. Moreover, this task moves towards the direction of automatic music arrangement (the same methodology could possibly be extended, in future, to more complex domains, such as voice or guitar or the whole song). Already now software like Melodyne [59], [60] delivers producers a powerful user interface to directly modify and adjust a spectrogram-based representation of audio signals to correct, perfect, reshape and restructure vocals, samples and recordings of all kinds. It is not unlikely that in the future artists and composers will start creating their music almost like they were drawing.

References

[1] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, *Deep learning techniques for music generation*. Springer, 2020.

[2] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue, "Computer-assisted composition at ircam: From patchwork to openmusic," *Computer music journal*, vol. 23, no. 3, pp. 59–72, 1999.

[3] A. Papadopoulos, P. Roy, and F. Pachet, "Assisted lead sheet composition using flowcomposer," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 2016, pp. 769–785.

[4] N. Jiang, S. Jin, Z. Duan, and C. Zhang, "Rl-duet: Online music accompaniment generation using deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 710–718.

[5] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "Popmag: Pop music accompaniment generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1198–1206.

[6] C. Lee, J. Shih, K. Yu, and H. Lin, "Automatic music genre classification based on modulation spectral analysis of spectral and cepstral features," *IEEE Transactions on Multimedia*, vol. 11, no. 4, pp. 670–682, 2009.

[7] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[8] M. Defferrard, S. P. Mohanty, S. F. Carroll, and M. Salathé, "Learning to recognize musical genre from audio," in *The 2018 Web Conference Companion*. ACM Press, 2018. [Online]. Available: https://arxiv.org/abs/1803.05337

[9] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[10] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Demucs: Deep extractor for music sources with extra unlabeled data remixed," *arXiv preprint arXiv:1909.01174*, 2019.

7. SUBJECTIVE EVALUATION

With an online study, we solicited 22 anonymous volunteers to rate the result for 3 out of 15 random drumless tracks (each 23.8 seconds) from the test split. Each time, a volunteer listened to a drumless tracks ($\mathbf{x}_*^{\mathrm{m}}$) first, and then (in random orders) the mixture (i.e., $\mathbf{x}_*^{\mathrm{m}} + \mathbf{x}_*^{\mathrm{d}}$) containing drum samples generated by four different models, plus the real human-made one (to set a high anchor). The volunteer then rated them in the following aspects on a 5-point Likert scale: **rhythmic consistency** between the drumless input and generated drums; **stylistic consistency** concerning the timbre and arrangement of the drumless input and generated drums; **audio quality** and **rhythmic stability** (whether the drummer follows a steady tempo) of the generated drum; and **overall** perceptual impression.

The mean opinion scores (MOS) in Table 1 show that `seq2seq+beat (low)` consistently outperforms the others, validating the effectiveness of using both the drumless codes and beat conditions. `decoder+beat (low)` performs consistently the second best, outperforming the two models without beat information significantly in three aspects according to paired t-test (p-value < 0.05), validating again the importance of the beat-aware module. Complementing Section 6, the MOS result suggests that the beat conditions seem more important than the drumless codes, though the best result is obtained with both.

Figure 5 further demonstrates that, given the same input, our model can generate multiple accompaniments with diversity in both beat and timbre. Diversity is an interesting aspect that is hard to evaluate, but it is desirable as there is no single golden drum accompaniment for a song. This may also explain why the F1 scores in Table 1 seem low.

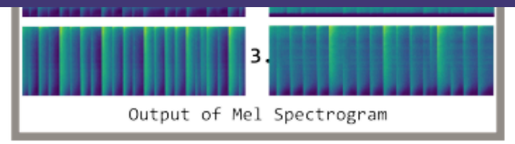


Figure 5: Two sets of three different generated samples by the same model given the same beat condition embedding.

Verbal feedbacks from the subjects confirm that our best model generates drum accompaniment that is rhythmically and stylistically consistent with the input, especially for band music or music with heavy use of bass. However, the model still has limits. At times the model generates total silence, though it can be avoided by sampling the LM again. The model may struggle to change its tempo going through different sections of a song. Moreover, the generation might be out-of-sync with the input in the beginning few seconds, until the model gets sufficient context. Please visit the demo page for various examples.

8. CONCLUSION

We have presented JukeDrummer, a novel audio-to-audio extension of OpenAI's JukeBox model capable of adding the drum part of a drumfree recording in the audio domain. To our knowledge, this represents the first attempt to audio-domain generation conditioned on drumless mixed audio. With objective and subjective evaluations, we validated the effectiveness of the customized VQ-VAE plus the seq2seq Transformer design, and the proposed beat-aware module. Among the beat conditions, we found that the low-level embeddings work the best. Future work can be done to further improve the language model (LM), and to extend our work to other audio-to-audio generation tasks.

JUKEDRUMMER: CONDITIONAL BEAT-AWARE AUDIO-DOMAIN DRUM ACCOMPANIMENT GENERATION VIA TRANSFORMER VQ-VAE

Yueh-Kao Wu
Academia Sinica
yk.lego09@gmail.com

Ching-Yu Chiu
National Cheng Kung University
x2009971@gmail.com

Yi-Hsuan Yang
Taiwan AI Labs
yhyang@ailabs.tw

I. The input data (MIDI-like events) represents drum hits as discrete symbols. Fine-grained timing variations (micro-timing, rubato, accelerando) are not explicitly captured.
As a result, the model may generate rigid patterns that fail to adapt to tempo fluctuations in different song sections.

Wu, Y.-K., Chiu, C.-Y., & Yang, Y.-H. (2022). JukeDrummer: Conditional Beat-Aware Audio-Domain Drum Accompaniment Generation via Transformer VQ-VAE. In Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR), Bengaluru, India.

Real-Time Drum Accompaniment Using Transformer Architecture

Behzad Haki*
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
behzad.haki@upf.edu

Marina Nieto*
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
marinanietogimenez@gmail.com

Teresa Pelinski
Centre for Digital Music
Queen Mary University of London
London, UK
t.pelinskiramos@qmul.ac.uk

Sergi Jordà
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
sergi.jorda@upf.edu

The ML Model uses a 2-bar attention span in the Transformer-based drum accompaniment model limits its ability to recognize long-term musical structures like verses and choruses, leading to repetitive, disjointed patterns. Since Transformers process input in fixed-length windows and lack recurrence, the model does not retain memory of previous sections.

Haki, B., Nieto, M., Pelinski, T., & Jordà, S. (2022). Real-Time Drum Accompaniment Using Transformer Architecture. In Proceedings of the 3rd Conference on AI Music Creativity (AIMC), Barcelona, Spain.

For this work, we focused on developing a real-time drum generator that continuously accompanies an instrumentalist. In this context, the generator is required to generate content that "supports" the instrumental performance. As a result, the generator needs to be continuously aware of the state of the performance, meaning that it needs to be not only aware of the performance near the time of generation, but also needs to be aware of the progression of the performance leading up to the time of generation. The transformer model developed for this work has an attention span of 2-bars. This means that the generative model in its current state is inherently incapable of taking into account the events received or generated prior to a given 2-bar segment used for generation. This limited attention span could potentially lead to fragmented and inconsistent generations and hence significantly limit the experience of the user with the system.

[7]Detailed analysis results: https://wandb.ai/mmil_upf/AIMC2022/reports/Analysis--VmlldzoyMzIyNjEx

During the design stage of the work, we were aware of this problem, however, as previously mentioned, the intention of this work was to investigate the effectiveness of a "bare minimum" transformer in a real-time accompaniment setting similar to the accompaniment setting shown in Figure 1. As such, we strictly decided to base our initial experiments on a short-term fixed span transformer, and if proven successful, in the future iterations incrementally expand the attention span and the memory of the network. To improve this issue, however, instead of modifying the network to have long-term memory, we decided to provide the model with a summary of past events memorized in the input buffer. Perhaps the simplest, yet highly effective, way of achieving this approach is through "over-dubbing" the previously played performed grooves, rather than providing the model with only two bars of performed groove leading up to the time of generation. With this approach, we can provide the model with a "sense" of the overall "feel" of the performance unfolding over a

GAP

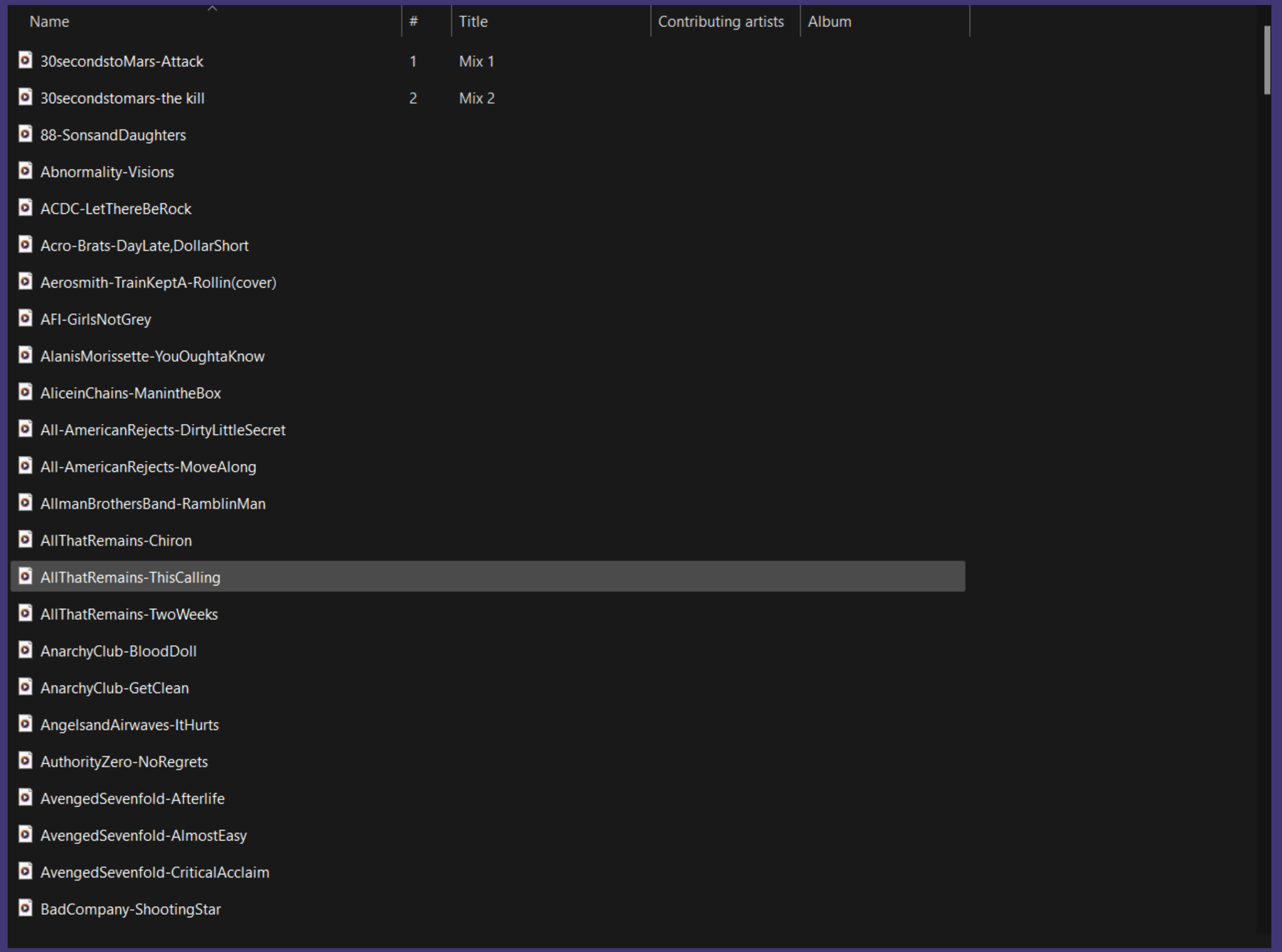
DRAWBACKS OF AI/ML MODELS

1. Models tend to give repetitive patterns.
2. Most models struggle to maintain long-term coherence and structural integrity in generated music.
3. Current research models display an inability to generate creative fills and dynamics which fit the structure of the song.
4. Models often perform poorly in complex genre's such as rock, metal, and jazz.
5. Many models require MIDI input, which represents music as discrete events without capturing continuous audio nuances, timbre, or symbolic expressions, leading to limitations in realism, timing accuracy, and musical depth.

DATASET COLLECTION & AUGMENTAION

710 rock/metal songs (.wav)

- Pitch Shifting: Shifted pitch by -2 to 2 semitones using librosa
- Time Stretching & Speed Variation: Altered playback speed by 0.8 to 1.2 times with pydub
- Partial Masking (SSM-specific): Randomly mask small sections of the Guitar SSM to encourage the model to infer missing rhythmic cues — useful for complex transitions.


Name
#
Title
Contributing artists
Album
30secondstoMars-Attack
1
Mix 1
30secondstomars-the kill
2
Mix 2
88-SonsandDaughters
Abnormality-Visions
ACDC-LetThereBeRock
Acro-Brats-DayLate,DollarShort
Aerosmith-TrainKeptA-Rollin(cover)
AFI-GirlsNotGrey
AlanisMorissette-YouOughtaKnow
AliceinChains-ManintheBox
All-AmericanRejects-DirtyLittleSecret
All-AmericanRejects-MoveAlong
AllmanBrothersBand-RamblinMan
AllThatRemains-Chiron
AllThatRemains-ThisCalling
AllThatRemains-TwoWeeks
AnarchyClub-BloodDoll
AnarchyClub-GetClean
AngelsandAirwaves-ItHurts
AuthorityZero-NoRegrets
AvengedSevenfold-Afterlife
AvengedSevenfold-AlmostEasy
AvengedSevenfold-CriticalAcclaim
BadCompany-ShootingStar

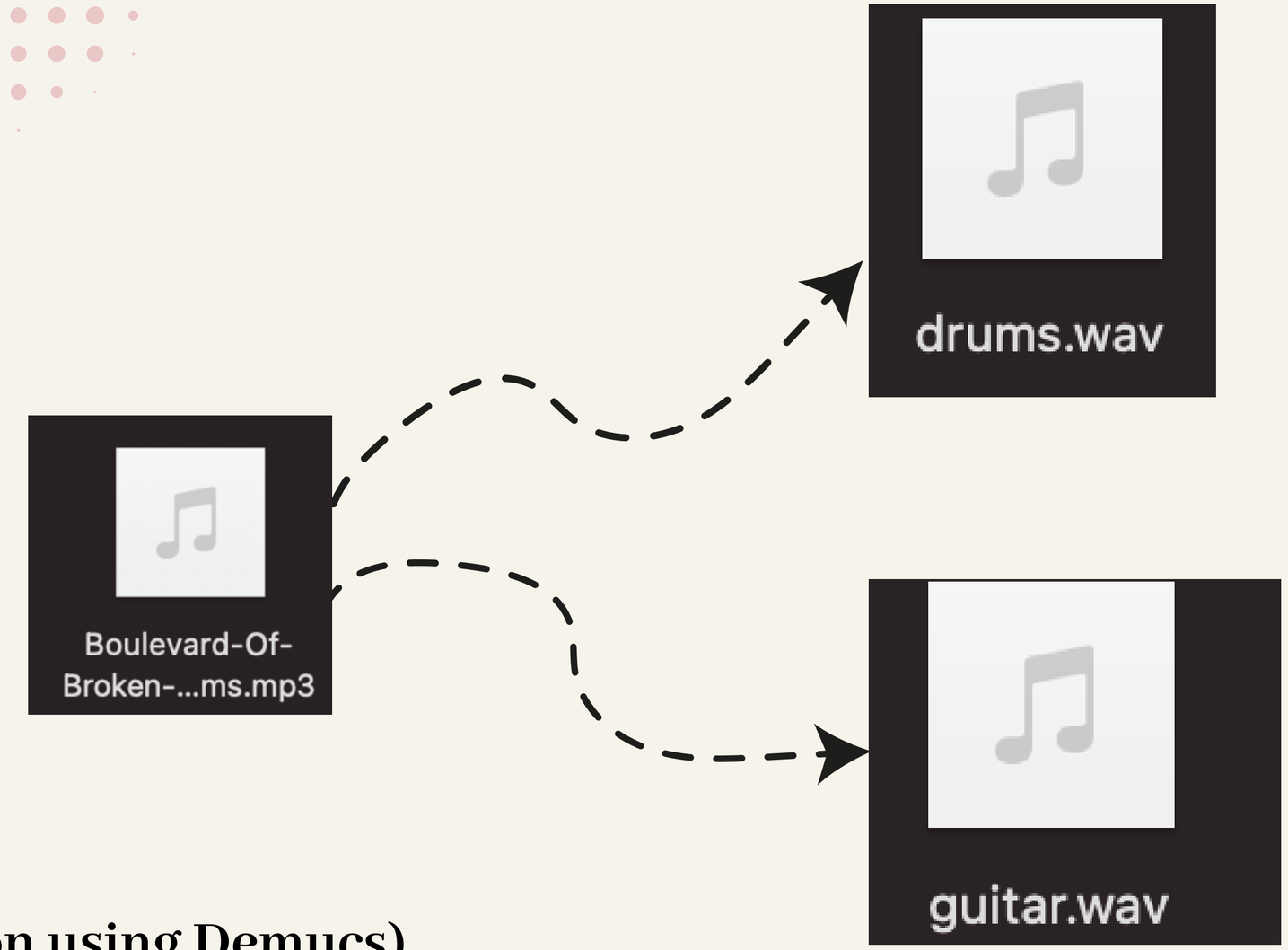
DATA PREPROCCESSING

Audio Preprocessing

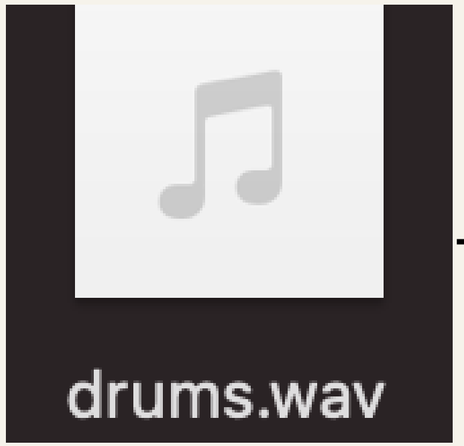
- **Audio separation into drum.wav and guitar.wav using demucs.**
- **Trimming Silence: If silence is present, remove silent parts to focus on relevant content, otherwise remove trailing part.**
- **Normalization: Ensure consistent audio amplitude levels.**

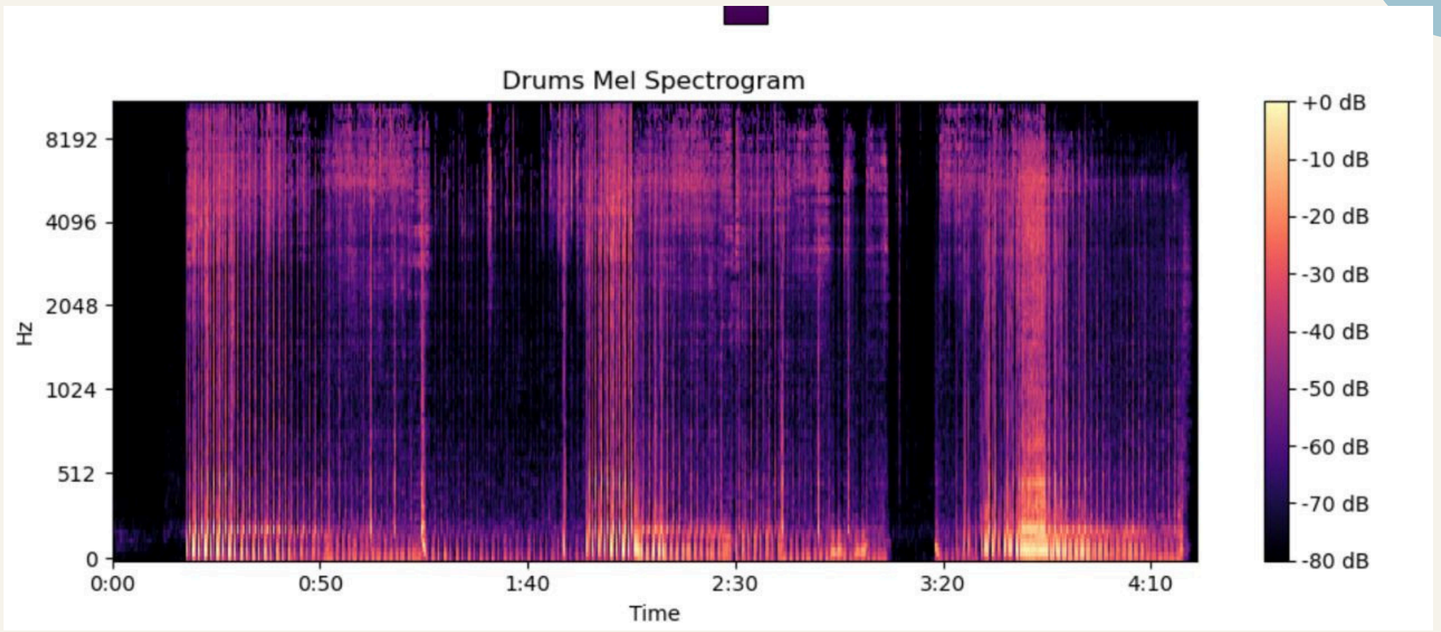
FEATURE EXTRACTION

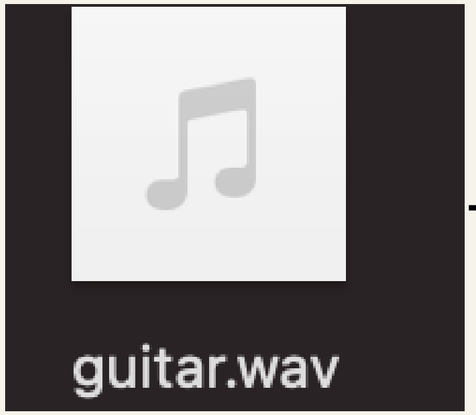
1. Mel Spectrogram Generation
   - Short-Time Fourier Transform (STFT): Converts audio to the frequency domain:
     - Instead of analyzing the entire signal at once, STFT applies a sliding window (2048 samples ≈ 46ms at 44.1kHz sample rate) across the signal.
     - Each frame is overlapping (512 samples step size/hop length) to maintain smooth transitions.
2. Self-Similarity Matrix (SSM) Computation
     - Calculate SSM: Measure similarity between Mel spectrogram frames
     - Using cosine similarity

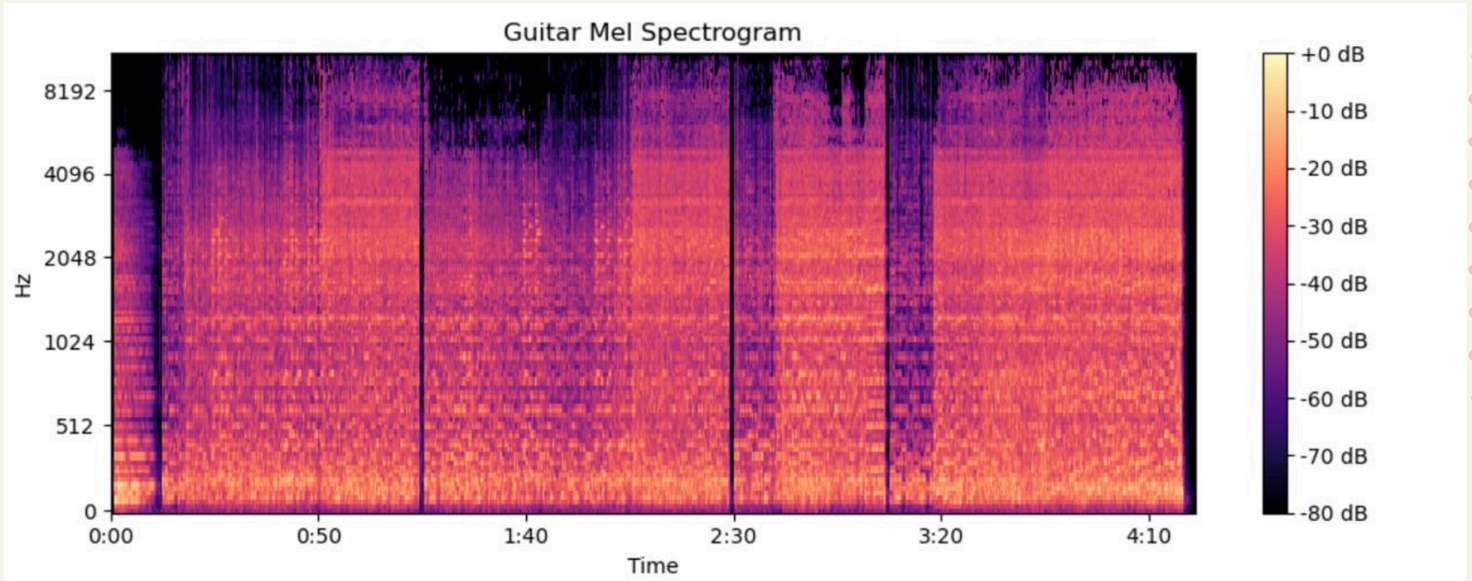


(Source separation using Demucs)

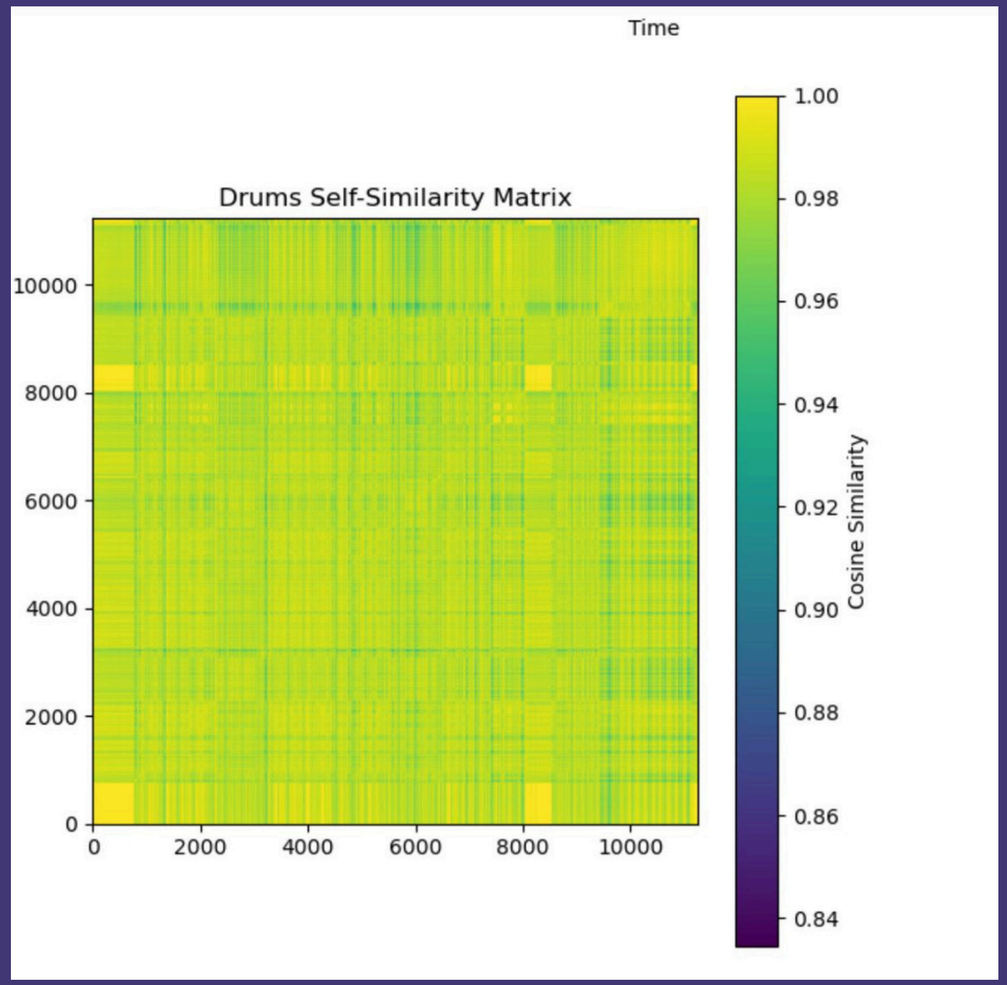

drums.wav

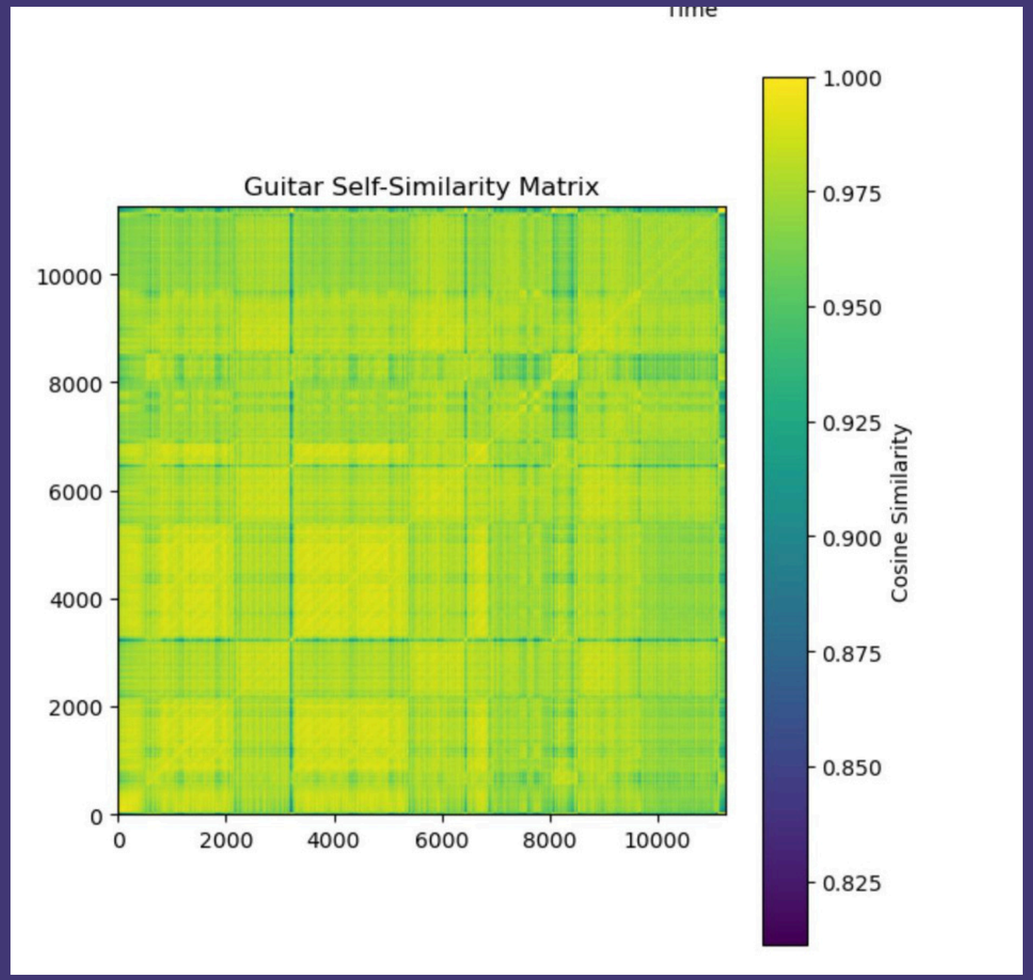

Drums Mel Spectrogram
8192
4096
2048
1024
512
0
Hz
0:00
0:50
1:40
2:30
3:20
4:10
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB


guitar.wav


Guitar Mel Spectrogram
8192
4096
2048
1024
512
0
Hz
0:00
0:50
1:40
2:30
3:20
4:10
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

SELF SIMILARITY MATRIX





SOLUTION ENHANCEMENT STRATEGY

1. Custom Dataset Creation: Extracts drum and guitar components from existing tracks to expand paired training data.
2. SSM Integration: Provides explicit structural information about repeated sections and transitions, ensuring generated drum patterns align with the song's structure.
3. Transformer Capabilities: Excels at capturing long-term dependencies in sequences, maintaining coherence across extended musical compositions.

ML MODELS

SSM+Transformer+Vocoder

EVALUATION

- MAE: Measures the average absolute difference between predicted and true values.
- MSE: Measures the average of squared differences, penalizing larger errors more heavily.
- Cosine Similarity: Measures the directional similarity between the predicted and true spectrogram vectors.
- SSIM: Measures perceptual similarity by comparing structural patterns and local contrasts in two images.

MODEL 1

SimpleTransformer

- Predicts drum Mel spectrogram and SSM from guitar Mel spectrogram and SSM
- Treats input as a 2-channel image
  - Channel 1 = Guitar Mel
  - Channel 2 = Guitar SSM
- Uses a shallow Transformer encoder over flattened image features

Data Handling

Input/Output Shapes

- Input → 2 × 128 × 128 (Guitar Mel + Guitar SSM)
- Output → 2 × 128 × 128 (Drum Mel + Drum SSM)

Preprocessing Steps

- Load full song Mel and SSM files
- Resize to 128 × 128 using adaptive_avg_pool2d
- Normalize each sample with min-max scaling

MODEL 1

SimpleTransformer

Limitations

No segmenting

→ Entire song compressed into a single 128×128 frame, aggressive downsampling may blur transients, fills, or quick changes

→ Loses temporal progression and rhythmic development

No modality-specific branches

→ Guitar Mel and SSM passed through a shared encoder

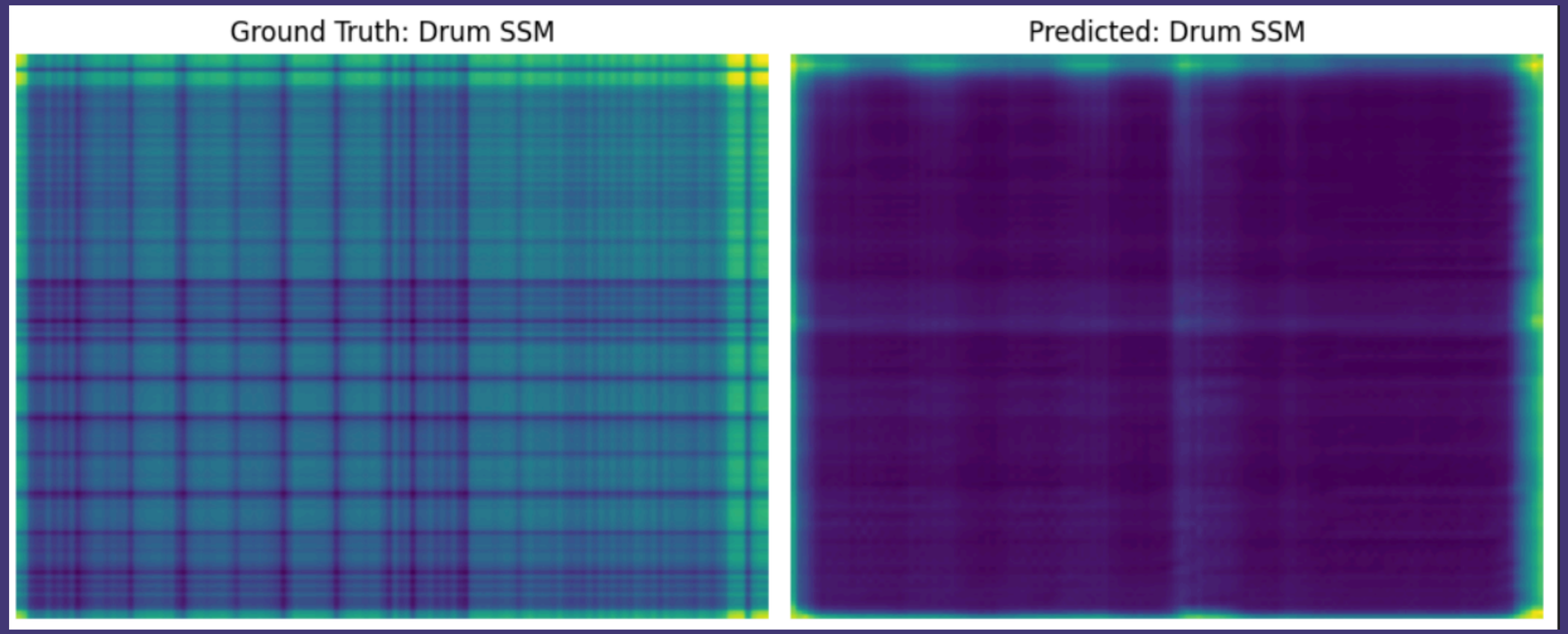
→ Fails to capture distinct structural vs spectral patterns

No positional encoding

→ Transformer receives tokens without time/frame context

→ Reduces ability to model global structure or rhythmic alignment

MODEL 1




```
Test MSE: 0.0324
Test MAE : 0.0983
Test Cosine Similarity : -0.1383
Test SSIM : 0.004
```

MODEL 2

DrumTransformer (Segment-wise, Dual-Encoder)

- Predicts drum Mel spectrogram and SSM from guitar Mel spectrogram and SSM
- Uses separate convolutional encoders for Mel and SSM
- Combines them using positional encoding
- Passes through a transformer encoder
- Outputs predicted Mel + SSM via two separate decoder heads

Data Handling (Preprocessing Steps)

- Segmented Input Strategy -Extract segments using a sliding window
- Each full-length song is split into overlapping segments:
  - 30-second intervals
  - 22-second stride
- ~1280-frame windows downsampled to 128 × 128 using adaptive_avg_pool2d
- Normalize each segment with min-max scaling

MODEL 2

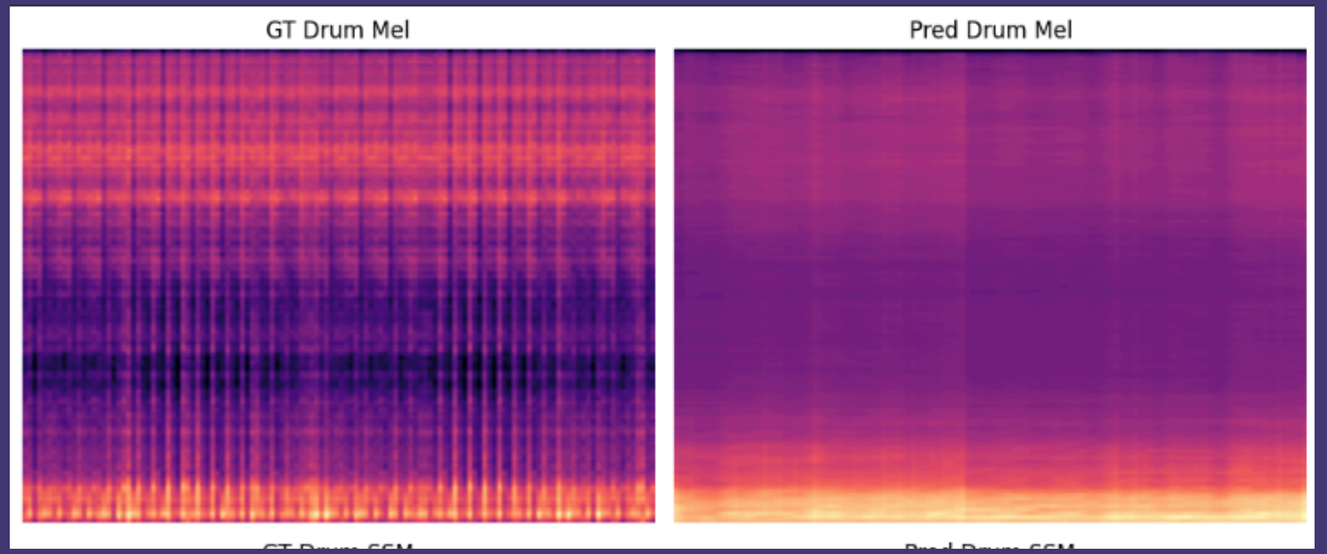
DrumTransformer (Segment-wise, Dual-Encoder)

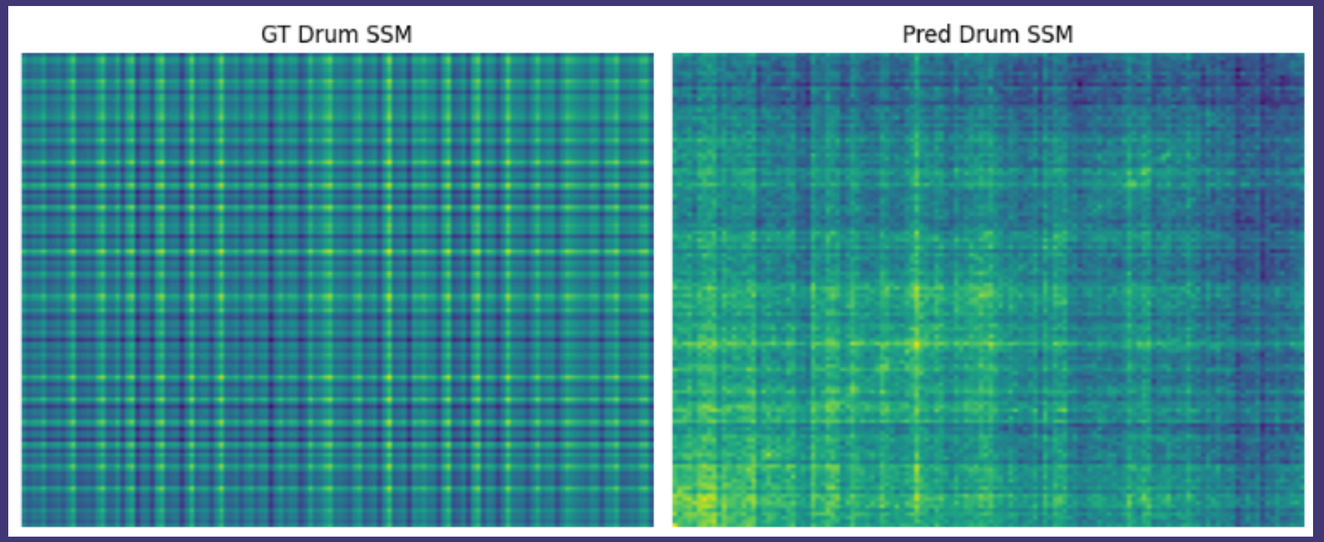
Limitations

- Fixed input resolution (128×128) still imposes compression → may blur fast drum events
- Adaptive pooling discards high-resolution structure → prevents the model from learning how to downsample
- Prediction compresses 30 seconds of audio into 128 frames → leads to temporal resolution loss
- No decoder recurrence — each segment is processed independently, ignoring global song context

MODEL 2

```
Test MSE: 0.124
Test MAE : 0.2054
Test Cosine Similarity : 0.216
Test SSIM : 0.0474
```





MODEL 3

DrumTransformer (Input mel - 128*T - 15s Segments, SSM-Guided)

- Predict the Mel spectrogram of the drum track of size (128*T) using:
  - Guitar's Mel spectrogram (128*T) → CNN → Tokens
  - Guitar's Self-Similarity Matrix (SSM) → Interp to 64×64 → CNN → Tokens
  - Concat → Positional Encoding → Transformer → Linear → Drum Mel (128×T)

Data Handling (Preprocessing Steps)

- Splits full-length tracks into:
  - 15-second segments (~128 frames)
  - With 11-second stride (~64 frames)
- Feeds each segment to the model one by one
- Improves training diversity by covering full song with overlap

MODEL 3

DrumTransformer (Input mel - 128*T - 15s Segments, SSM-Guided)

SSM Downsampling via Interpolation

- Each 15 secs SSM → downsampled to 64×64
- Uses F.interpolate with bilinear mode
- Provides structural cues to the mel spectogram while cutting computation

Dual CNN-Based Encoders

- Separate encoders for:
  - Guitar Mel spectrogram
  - Guitar SSM
- Learns modality-specific features:
  - Local spectral patterns (Mel)
  - Structural repetition and form (SSM)

MODEL 3

DrumTransformer (Input mel - 128*T - 15s Segments, SSM-Guided)

Transformer with Positional Encoding

- Outputs from both encoders are:
  - Flattened → Tokenized
  - Concatenated → Combined into one sequence
- Uses sinusoidal positional encoding
- Passed to Transformer Encoder for joint modeling of:
  - Temporal progression (Mel)
  - Structural alignment (SSM)

CHALLENGES

- Limited Paired Dataset
  - The availability of cleanly paired guitar and drum tracks was limited, restricting the model’s generalization capacity.
- Manual preprocessing (e.g., Mel spectrograms, SSMs) further reduced usable data due to noise or alignment issues.
- High Memory Footprint of Transformers
  - Processing full-resolution spectrograms (128×T) and especially SSMs (T×T, e.g., 1250×1250) consumed excessive GPU memory.
  - Standard self-attention scales quadratically with input size, limiting batch sizes and input length.

CHALLENGES

- Segmented Training Trade-offs
  - Dividing audio into smaller chunks (e.g., 128×128 Mel frames) helped reduce memory usage but introduced discontinuities at segment boundaries, harming temporal consistency.
- Inefficient SSM Encoding
  - Patch-based encoding for SSMs caused memory overflows on GPU.
  - Switching to interpolation-based downsampling helped but reduced structural detail fidelity.
- Long Training Times
  - Due to large input sizes and model complexity, training even for a few epochs required long runtimes, discouraging iterative experimentation.

DEPLOYMENT

- Jam Assistance: Automatically adds expressive drum tracks to solo guitar performances, making solo jamming feel like a full-band experience.
- Practice: Generate dynamic drum tracks for practice sessions.
- Give the drummer a basic groove to build on when there's a creative block.
- Songwriting Aid: Provides a foundational drum layer to help composers sketch new song ideas quickly and intuitively.
- Can be deployed as a web app, Jupyter interface, or DAW plugin for ease of access.

SCALING CHALLENGES

- High GPU memory usage due to large Mel and SSM input sizes (e.g., 128×T, T×T).
- Splitting songs into overlapping segments increases inference time.
- Real-time inference is not yet supported; model optimization may be needed.
- Limited dataset generalization — the model is trained on clean rock/metal tracks and may underperform on noisy or genre-diverse input.
- Handling large .npy files for Mel and SSM features at scale can lead to significant disk, RAM, and I/O bandwidth pressure.

THANK YOU