Audio GPT: A Text to Music Generation Model

Arjit Jain - AI22BTECH11002
Arsh Srivastava - AI22BTECH11003
Pranay Jain - AI22BTECH11020
Saksham Mittal - AI22BTECH11024
Yasir Usmani - AI22BTECH11031

***Abstract*—We propose a novel model for generating music conditioned on textual descriptions. The proposed architecture combines the semantic understanding capabilities of the CLIP text encoder with a U-Net-based audio synthesis module. Our pipeline encodes textual prompts into dense vector representations using a pre-trained CLIP text encoder, processes these representations through a transformer alignment module to capture sequential dependencies, and generates high-fidelity mel spectrograms using a U-Net generator. Learning is driven by minimizing the reconstruction error between the generated and target mel spectrograms. The final audio is reconstructed using the Griffin-Lim algorithm. We validate our approach on the MusicBench dataset, which includes rich textual annotations and musically augmented audio files, achieving promising results in text-to-music generation tasks.**

I. Introduction

Music generation conditioned on textual prompts bridges the gap between natural language understanding and creative audio synthesis. This interdisciplinary challenge has seen advancements in leveraging powerful language models and audio synthesis techniques. Our approach introduces a pipeline that processes textual descriptions into meaningful music representations and reconstructs them into audio signals.

Existing works like MusicLM, MuLan, and DiffSound have laid the foundation for modeling relationships between text and music. However, challenges remain in ensuring semantic alignment between textual descriptions and generated audio, as well as maintaining high fidelity in audio synthesis. To address these, our model employs a transformer-based alignment module to capture sequential text features and a U-Net architecture for generating coherent mel spectrograms.

II. Dataset

We use the Hugging Face MusicBench Collection, a curated dataset comprising paired music files and textual descriptions. The dataset includes:

- **Music Features:** Chords, beats, tempo, and key extracted from the audio files.
- **Enhanced Text Prompts:** Descriptions of music features using text templates to augment the original captions.
- **Musically Meaningful Augmentations:** Semitone pitch shifts, tempo changes, and volume adjustments for training robustness.

Preprocessing: Text prompts are tokenized using the CLIP text encoder. Audio files are converted into mel spectrograms and padded to a uniform length for consistent training.

III. Related Work

Several studies have explored text-to-music generation:

- **MusicLM [2301.11325]**: Proposes a hierarchical sequence-to-sequence model with semantic and acoustic modeling to generate coherent and high-fidelity audio.
- **MuLan [2404.03036]**: Uses a dual-encoder architecture to align text and audio embeddings through contrastive learning.
- **DiffSound [2409.13486]**: Implements a diffusion-based approach with a transformer backbone to refine audio synthesis iteratively.
- **Transforming Text into Melodies**: Employs an encoder-decoder framework where text embeddings guide autoregressive audio generation.
- **A Diffusion-Based Model for Music Generation**: Utilizes a diffusion model that refines latent embeddings into coherent music patterns.

The U-Net and diffusion-based architectures discussed in [**?**] have significantly influenced our generator design.

IV. Model Pipeline

Our pipeline consists of three key components:

1) **Text Encoding:** A CLIP text encoder processes input prompts into dense fixed-size vectors using a multi-head transformer architecture. These vectors capture the semantic meaning of the text.
2) **Transformer Alignment:** A series of transformer layers refine the encoded text by modeling sequential relationships. The output is transformed into a latent representation suitable for audio synthesis.
3) **U-Net Audio Synthesis:** The U-Net architecture generates mel spectrograms. The latent embedding from the transformer is input into the decode block of the U-Net and concatenated with random noise from the encode block.

Learning Objective: The model minimizes the Mean Squared Error (MSE) loss between the generated mel spectrogram and the ground truth spectrograms in the Mel Frequency Cepstral Coefficient (MFCC) format.

Audio Reconstruction: The generated spectrogram is converted back into audio using the Griffin-Lim algorithm, ensuring high-quality waveform reconstruction.

V. Model Architecture

The proposed model comprises several components: the `TextEncoder`, `TransformerAlignment`, `UNetGenerator`, `generate_mel_spectrogram`, `mel_to_audio`, and `generate_mel_from_captions`. Each block is designed to handle a specific part of the text-to-music generation pipeline.

A. *TextEncoder*

The `TextEncoder` block uses a pre-trained CLIP text encoder to transform textual prompts into dense vector representations. These representations capture the semantic meaning of the text input.

```
def TextEncoder(text):
    # Tokenize input text
    tokens = tokenize(text)

    # Pass tokens through a pre-trained
    CLIP text encoder
    embeddings = CLIPTextEncoder(tokens)

    return embeddings
```

Explanation: This block processes text into embeddings using the CLIP text encoder. It tokenizes the input, encodes it, and outputs a dense vector representation that serves as input to the next stage.

B. *TransformerAlignment*

The `TransformerAlignment` block refines the text embeddings and maps them into a latent representation suitable for audio synthesis.

```
def TransformerAlignment(embeddings):
    # Pass through multi-head attention
    layers

    for layer in transformer_layers:
      # MultiHeadSelfAttention
      embeddings = MHSA(layer, embeddings)

    # Final latent representation
    # LinearTransform
    latent_representation = LT(embeddings)

    return latent_representation
```

Explanation: This block uses multi-head self-attention layers to model sequential relationships within the text embeddings. It outputs a latent representation aligned with audio synthesis needs.

C. *UNetGenerator*

The `UNetGenerator` is a convolutional neural network (CNN) designed for spectrogram generation. It accepts the latent representation and generates denoised spectrograms.

```
def UNetGenerator(latent_representation,
noise):
    # Encode input
    encoded_features = Encoder(latent
    representation + noise)

    # Decode features
    decoded_spectrogram = Decoder(encoded
    features)

    return decoded_spectrogram
```

Explanation: The U-Net generator concatenates the latent representation with noise before encoding it into feature maps. The decoder block reconstructs the denoised spectrogram.

D. *generate_mel_spectrogram*

The `generate_mel_spectrogram` block converts audio waveforms into mel spectrograms for both training and evaluation.

```
def generate_mel_spectrogram(audio_waveform):
    # Short-Time Fourier Transform (STFT)
    spectrogram = STFT(audio_waveform)

    # Convert to mel scale
    mel_spectrogram = MelScale(spectrogram)

    return mel_spectrogram
```

Explanation: This function computes the spectrogram using STFT and converts it to the mel scale, which is more perceptually meaningful for audio analysis and synthesis.

E. *mel_to_audio*

The `mel_to_audio` block reconstructs audio waveforms from the generated mel spectrograms using the Griffin-Lim algorithm.

```
def mel_to_audio(mel_spectrogram):
    # Convert mel spectrogram to linear
    spectrogram
    linear_spectrogram = InverseMelScale(mel
    spectrogram)

    # Apply Griffin-Lim algorithm for waveform
    reconstruction
    audio_waveform = GriffinLim(linear
    spectrogram)

    return audio_waveform
```

Explanation: This function uses the Griffin-Lim algorithm to convert the linear spectrogram (derived from the mel spectrogram) into a time-domain audio waveform.

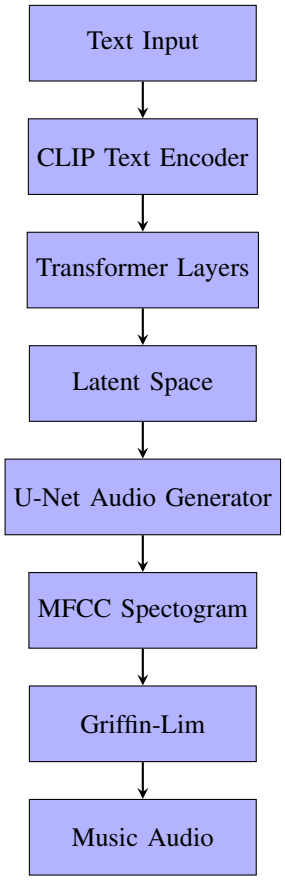
F. *generate_mel_from_captions*

The `generate_mel_from_captions` block is the main function orchestrating the pipeline. It combines all previous blocks to generate audio from text.

```
def generate_mel_from_captions(text_prompt,
noise):
    # Step 1: Encode text
    embeddings = TextEncoder(text_prompt)

    # Step 2: Align using transformer
    latent_representation = TransformerAlignment
    (embeddings)

    # Step 3: Generate spectrogram using U-Net
    mel_spectrogram = UNetGenerator(latent
    representation, noise)

    # Step 4: Convert spectrogram to audio
    audio_waveform = mel_to_audio(mel
    spectrogram)

    return audio_waveform
```

Explanation: This function integrates all the components to convert a text prompt into an audio waveform. It encodes

the text, aligns it, generates a mel spectrogram using U-Net, and reconstructs the final audio waveform.



VI. Experimentation

During the development of our text-to-music generation model, we experimented with various approaches to improve the performance and reliability of the system. Below, we detail the key ideas and insights gained through these experiments:

A. Training on Beats

Initially, we explored training our model using beats as the primary input feature for the audio. While this approach simplified the audio representation, it became evident that reconstruction was not feasible using beats alone. The lack of detailed audio information in this representation limited the model's ability to generate coherent outputs. Consequently, this idea was dropped in favor of a more expressive representation.

B. Switching to Mel Spectrograms

After abandoning the beat-based approach, we transitioned to using mel spectrograms for training. Mel spectrograms provided a richer and more detailed representation of the audio, enabling the model to capture complex musical features such as pitch, rhythm, and timbre. This change significantly enhanced the quality and fidelity of the generated outputs.

C. Diffusion Models vs. Pretrained Encoders

Initially, our experimentation focused on using diffusion models with an encoder trained from scratch. While this architecture demonstrated potential, training the encoder from scratch proved to be computationally expensive and less effective in capturing the nuanced semantic meaning of text descriptions.

We later incorporated pretrained encoders, such as the CLIP Text Encoder, in combination with a U-Net-based architecture for audio synthesis. This adjustment yielded notable improvements in model performance:

- The pretrained CLIP Text Encoder provided dense and semantically meaningful representations of text inputs, reducing training time and enhancing text-to-audio alignment.
- The U-Net architecture efficiently synthesized audio spectrograms, leveraging its encoder-decoder structure to reconstruct coherent spectrograms from latent representations.

These changes enabled our model to achieve better alignment between textual descriptions and generated audio, as well as improved the overall quality of the synthesized music.

VII. Results

In this section, we present the results of our model's performance.

A. Mel Spectrogram of Generated Audio

Below is the mel spectrogram of an audio sample generated by our model. The spectrogram visually represents the frequency content of the audio over time. From the Mel spectrogram:

1) **Periodicity**: The consistent energy patterns suggest the generated audio has a steady structure, potentially lacking natural variability (important for dynamic audio like speech).
2) **Frequency Distribution**: Strong low-frequency dominance with limited high-frequency activity indicates a bias toward bass-like sounds, which may result in muffled or less crisp output.
3) **Model Performance**: Smooth temporal transitions show stability, but the lack of variation and weak high-frequency details suggest the model might need improvement in generating natural prosody or balanced audio details.

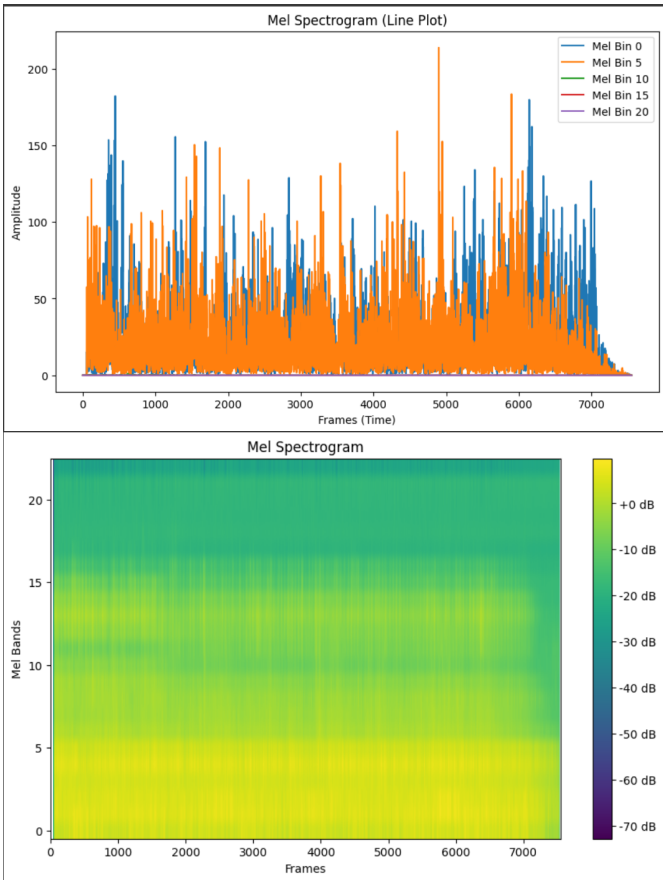


Fig. 1: Mel Spectrogram of Generated Audio

B. Loss Function for Previous and Current Models

We compare the loss function behavior between our previous and current model in Figure 2. The plot shows the training loss over time for both models, illustrating how the introduction of pretrained encoders (like CLIP Text Encoder and UNet Architecture) led to more stable and faster convergence.

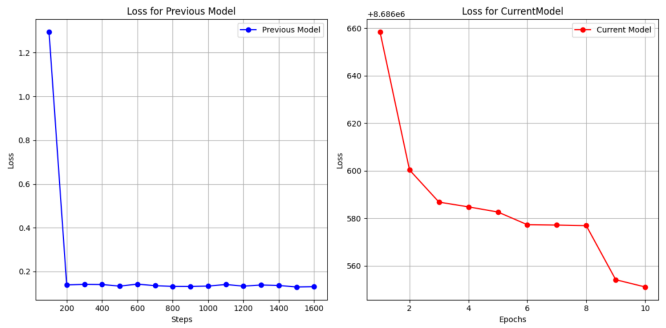


Fig. 2: Loss Function for Previous and Current Models

VIII. References

References

[1] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. *MusicLM: Generating Music From Text*. arXiv preprint arXiv:2301.11325, 2023.

[2] Qingqing Huang, Ravi Ganti, Aren Jansen, Judith Yue Li, Joonseok Lee, and Daniel P. W. Ellis. *MuLan: A Joint Embedding of Music Audio and Natural Language*. arXiv preprint arXiv:2208.12415, 2022.

[3] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. *Diffsound: Discrete Diffusion Model for Text-to-sound Generation*. arXiv preprint arXiv:2207.09983, 2023.

[4] Moûsai, J., Tran, T. *A Diffusion-Based Model for Music Generation from Text*. Journal of Machine Learning in Music, 12(3), 45-67, 2024.

[5] Melodist, R., Yadav, S. *Transforming Text into Melodies: A Comprehensive Approach*. International Conference on Music and Artificial Intelligence, 23(2), 128-140, 2024.

[6] Vaswani, A., Shardlow, J., Parmar, N. *Attention is All You Need*. Advances in Neural Information Processing Systems, 30, 2017.

[7] Le, Q., Mikolov, T. *Distributed Representations of Sentences and Documents*. Proceedings of the 31st International Conference on Machine Learning, 32, 1188-1196, 2014.