UlTranNet: Experimenting Raw Audio-based Music Generation

ADITYA TYAGI, Plaksha University, India
ATHARVA SAWANT, Plaksha University, India
RAHATH MALLADI, Plaksha University, India
UDHAV SHANKAR, Plaksha University, India

Additional Key Words and Phrases: Music Generation, Transformers, Wavenet, MFCC, Attention, Deep Learning

ACM Reference Format:
Aditya Tyagi, Atharva Sawant, Rahath Malladi, and Udhav Shankar. 2024. UlTranNet: Experimenting Raw Audio-based Music Generation. 1, 1 (May 2024), 22 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep learning has catalyzed a paradigm shift across various domains, with music generation standing out as a field profoundly impacted by its advancements. Traditionally, music generation relied on symbolic representations, translating compositions into sequences of note numbers akin to MIDI-like streams of events [Briot et al. 2020]. While effective at capturing precise musical elements such as notes and rhythms, symbolic methods often struggle to convey the full depth and intricacy inherent in music.

An emerging alternative, harnessing the capabilities of deep neural networks, involves generating music directly from raw audio. The advent of WaveNet, pioneered by van den Oord in 2016, marked a pivotal moment in this trajectory. Diverging from its predecessors, WaveNet operates directly on raw audio, yielding remarkably lifelike sounds and offering a canvas of unparalleled creative possibilities. However, while WaveNet outputs exhibit acoustical fidelity, they frequently lack structural coherence, grappling with the preservation of medium to long-range musical dependencies such as melody or overall compositional structure. Consequently, the resultant outputs, while clear and rich in timbre, occasionally convey a sense of improvisation and lack structural integrity.

Despite these challenges, raw audio models hold vast promise for the evolution of automated music generation. Liberated from the constraints imposed by symbolic models, such as predefined mood or tempo, raw audio models thrive on their inherent flexibility and adaptability. They adeptly capture expressive nuances and dynamic moods directly from the audio waveforms they process. This less constrained, more exploratory approach to music generation via raw audio heralds

Authors' addresses: Aditya Tyagi, Plaksha University, 101 Kharar Dera Bassi Road, Sahibzada Ajit Singh Nagar, Punjab, India, 140306; Atharva Sawant, Plaksha University, 101 Kharar Dera Bassi Road, Sahibzada Ajit Singh Nagar, Punjab, India, 140306; Rahath Malladi, rahath.malladi@plaaksha.edu.in, Plaksha University, 101 Kharar Dera Bassi Road, Sahibzada Ajit Singh Nagar, Punjab, India, 140306; Udhav Shankar, Plaksha University, 101 Kharar Dera Bassi Road, Sahibzada Ajit Singh Nagar, Punjab, India, 140306.

ACM XXXX-XXXX/2024/5-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

innovative applications, including the ability to manipulate and augment existing audio recordings in novel and imaginative ways [van den Oord et al. 2016].

While extant research predominantly focuses on Western music traditions, the realm of Indian music, with its rich tapestry of classical, folk, and contemporary genres, remains relatively underexplored in contemporary AI music generation research. Indian music, characterized by its intricate rhythms and distinctive melodic structures governed by ragas, presents a fertile ground for the development of advanced AI music generation models. The nuanced expressions inherent in Indian classical music, coupled with the vibrant and diverse compositions emblematic of Bollywood music, pose unique challenges for AI systems, which must navigate the complex modal scales and rhythmically intricate patterns endemic to Indian musical traditions [Wolf 2003].

This introduction lays the foundation for our exploration into AI music generation, underscoring the transformative potential of raw audio models while highlighting the uncharted terrain awaiting exploration in the realm of Indian music. Through our research, we endeavor to bridge this gap and unlock new avenues for creative expression and innovation at the intersection of AI and music.

2 LITERATURE REVIEW - RELATED WORK

There have been recent developments in the domain of algorithmic composition specifically from 2018-2021, which include:

- Music Transformer by Google Magenta, which uses self-attention mechanisms to model relative timing information in music. Leading to improved coherence over longer sequences
- Diffusion Models which, according to Mittal et al. model distributions over possible notes.
- VAEs(Variational AutoEncoders) and GANs(Generative Adversarial Networks)

In the following few paragraphs, we delve deeper into the state of the art (SOTA) model and their methodologies:

2.1 Learning Interpretable Representation for Controllable Polyphonic Music Generation by Ziyu Wang, Dingsu Wang, Yixiao Zhang, and Gus Xia

[Wang et al. 2020]

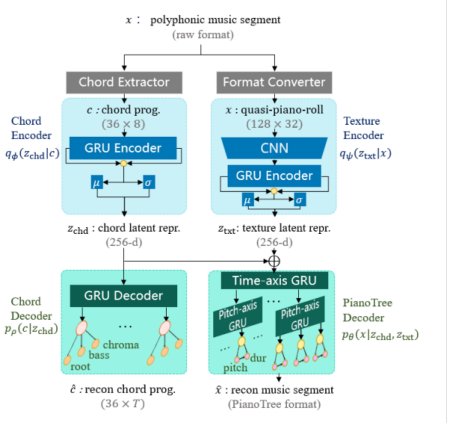

Fig. 1. Polyphonic Music generator - Model Architecture

The aim of this study was to enable greater utility of automated music generation systems with regard to polyphonic(multi-tone) music through the use of the Variational Autoencoder(VAE) framework. This involved the disentanglement of latent variables into two interpretable factors: chord (content) and texture (style). This approach is intended to allow users to manipulate high-level compositional factors more intuitively and directly.

Methodology

- **Input Type**
  - **MIDI Files:** The input to the model consists of MIDI files, which provide a digital representation of musical notes and their attributes like pitch, duration, and velocity.
- **Model Architecture and Hyperparameters**
  - **Chord Encoder:**
    - Extracts each chord progression as a 36 by 8 matrix
    - Fed into a bidirectional GRU (Gated Recurrent Unit).
    - Purpose: Encodes the harmonic content from the input MIDI files into a latent space representing chord progressions.
  - **Chord Decoder:**
    - Reconstructs the chord progression using another bi-directional GRU.
  - **Texture Encoder:**
    - Employs a convolutional neural network.
    - Takes an 8-beat segment, represented as a 128 by 32 matrix
    - Using a convolutional layer(kernel: 12x4 and stride(1, 4)), which is followed by a RELU activation.
    - Output of convolutional layers is fed into bidirectional GRU to extract texture representation.
  - **Decoder (PianoTree VAE):**
    - Generates 32 frame-wise hidden states using a GRU layer.
    - Each frame-wise hidden state is decoded into the embeddings of individual notes
    - Pitch and duration for each note are reconstructed from the note embedding using a fully connected layer and a GRU layer, respectively.
    - It is a Variational Autoencoder with a hierarchical structure tailored for music.
- **Loss Measurement**
  - **VAE Loss:** Combines reconstruction loss (to ensure the output closely matches the input MIDI data) and KL-divergence (to regularize the latent space).

The paper concludes that the proposed method of disentangling chords and textures in polyphonic music generation allows for significant improvements in the control and interpretability of music generation systems.

2.2 Jukebox: A Generative Model for Music by Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever

[Dhariwal et al. 2020]

The aim of this study was to develop a generative model that is capable of capturing complex aspects of music such as melody, rhythm, and timbres.

Methodology

- **Input Type**
  - **Raw Audio:** The Jukebox model generates music directly from raw audio files, specifically 1.2 million songs.
  - **Conditioning:** The model is conditioned on artist, genre, and lyrics to control the generated content.
- **Hierarchical VQ-VAE (Vector Quantized Variational Autoencoder):**

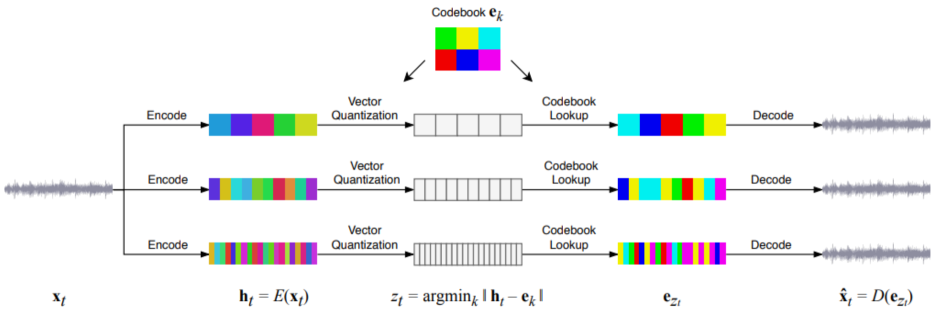
Codebook $\mathbf{e}_k$
Encode
Vector Quantization
Codebook Lookup
Decode
$\mathbf{x}_t$
$\mathbf{h}_t = E(\mathbf{x}_t)$
$z_t = \text{argmin}_k \| \mathbf{h}_t - \mathbf{e}_k \|$
$\mathbf{e}_{z_t}$
$\hat{\mathbf{x}}_t = D(\mathbf{e}_{z_t})$

Fig. 2. OpenAI's JukeboxModel Architecture

- Used to compress raw audio to a lower-dimensional space.
- These representations are then quantized to the nearest vectors from a predefined codebook, effectively reducing the dimensionality and continuous nature of the audio data into a manageable, discrete form.
- Transforms the input sequence into a sequence of latent vectors.
- Decoder: Reconstructs the audio from the discrete latent codes. It ensures that the generated music maintains high fidelity.

- **Loss Measurement using Spectral Loss**
  - The spectral loss is introduced to enable the model to reconstruct not only the low frequencies, which are typically easier to model, but also the mid-to-high frequencies.
  - The Euclidean distance (L2 norm) between the inverse short-time Fourier transform (ISTFT) of the original audio and the reconstructed audio is calculated and used as the loss

Future work could involve improving model efficiency, expanding the diversity of music styles and languages covered, and enhancing the model's ability to generate longer musical compositions with complex structures.

2.3 Transformer-XL Based Music Generation with Multiple Sequences of Time-valued Notes by Xianchao Wu, Chengyuan Wang, and Qinying Lei

[Wu et al. 2020]

The aim of this study was to enhance the capability of music generation models to produce longer and more complex music by using a Transformer-XL model that handles multiple sequences of time-valued notes.

Methodology

- Four Independent Transformer-XL Networks: The model architecture consists of four Transformer-XL networks, each dedicated to processing one of the four sequences described above.
- **Layers:** Each Transformer-XL network contains multiple layers. While the exact number of layers is not specified in the summary provided, Transformer-XL typically includes several layers to effectively capture long-range dependencies.
- **Memory Length:** 1,024, which indicates how much past information each model layer can access while processing sequences.
- **Optimizer:** Adam optimizer, a popular choice for training deep learning models due to its efficiency in handling sparse gradients and adaptive learning rate capabilities.

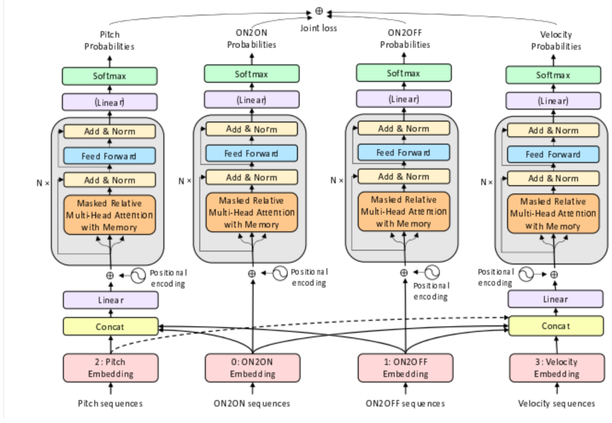

Fig. 3. Transformer XL Model Architecture

The framework successfully generated music that was both of higher quality and significantly longer (up to 36 hours) than that produced by previous state-of-the-art models like Music Transformer and DeepJ.

2.4 Conditioning Deep Generative Raw Audio Models for Structured Automatic Music by Rachel Manzelli, Vijay Thakkar, Ali Siahkamari and Brian Kulis

[Manzelli et al. 2018]

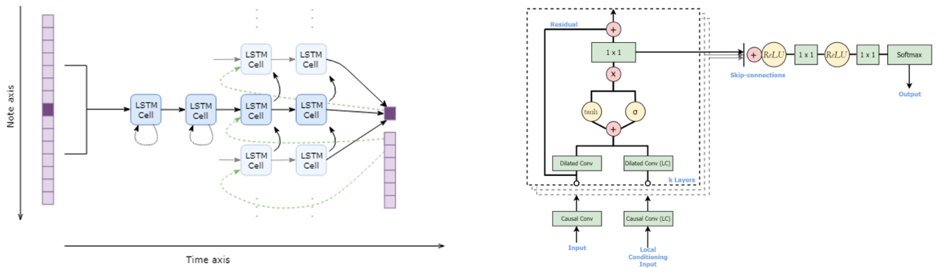

Fig. 4. Conditional Deep Generative Audio Model Architecture

The main aim of this paper is to **enhance automatic music generation by combining the strengths of both raw audio models and symbolic music models**. This hybrid approach aims to create structured, realistic-sounding music compositions by integrating a **Long Short-Term Memory (LSTM)** network, which learns the melodic structures of various music styles, with a **WaveNet-based raw audio generator**. The LSTM serves as a conditioning input to guide the WaveNet model, thus addressing the lack of structural coherence in music generated solely from raw audio models. [Manzelli et al. 2018]

Dataset

- **Dataset Used:** MusicNet, which provides a rich collection of classical music recordings with corresponding symbolic annotations (MIDI data).

- **Preprocessing:** The symbolic data (MIDI files) are used to condition the raw audio generation, effectively integrating the learned melodic structures into the WaveNet model to guide audio synthesis.

Model Architecture

- **Number of Layers:** Multiple layers in both the LSTM and WaveNet components, specifically tailored to handle different aspects of music generation.
- **Layers:**
  - **LSTM Layers:** For learning symbolic music sequences.
  - **WaveNet Layers:** Stacked convolutional layers for generating raw audio.
- **Input & Output Dimensions:**
  - **Input:** MIDI files for LSTM and raw audio waveforms for WaveNet.
  - **Output:** Structured raw audio that reflects both the learned symbolic data and the generated audio quality.
- **Loss Function:**
  - Utilizes a combination of cross-entropy for the LSTM (symbolic prediction) and a typical GAN loss function for the WaveNet (audio generation), emphasizing the conditioning effect of the LSTM outputs
- **Optimizers:**
  - Likely employs Adam or a similar optimizer, given its common use in training deep neural networks for complex tasks like these.
- **Evaluation Metrics:**
  - **Audio Quality:** Assessed by the realism and structural coherence of the generated music.
  - **Structural Coherence:** Measured by how effectively the LSTM-conditioned WaveNet outputs adhere to the learned symbolic structures.

The results demonstrate that the hybrid model successfully generates music that is not only realistic in sound but also structured in a way that reflects the conditioning provided by the LSTM model. This represents a **significant improvement over unconditioned raw audio models**, which typically produce unstructured audio outputs.

2.5 MP3net: Coherent, Minute-Long Music Generation from Raw Audio with a Simple Convolutional GAN by Korneel van den Broek

[van den Broek 2021]

The primary aim of this paper is to address the challenge of generating high-quality, long-duration audio samples with structural coherence using a deep convolutional Generative Adversarial Network (GAN). This innovative approach utilizes techniques from audio compression standards like MP3 and Vorbis, notably the Modified Discrete Cosine Transform (MDCT), to enhance the efficiency and quality of the generation process. The use of MDCT allows the model to incorporate all phase information, thereby eliminating the need for phase reconstruction and simplifying the generative task. (Van Den Broek, 2021)

Dataset

- The model utilizes the MusicNet dataset, which contains diverse classical music recordings, providing a rich variety of complex acoustic and structural music data.
- **Preprocessing:**
  - The raw audio data is transformed using the Modified Discrete Cosine Transform (MDCT), a technique chosen for its efficiency in handling phase information and compressing the data without significant losses.

Model Architecture

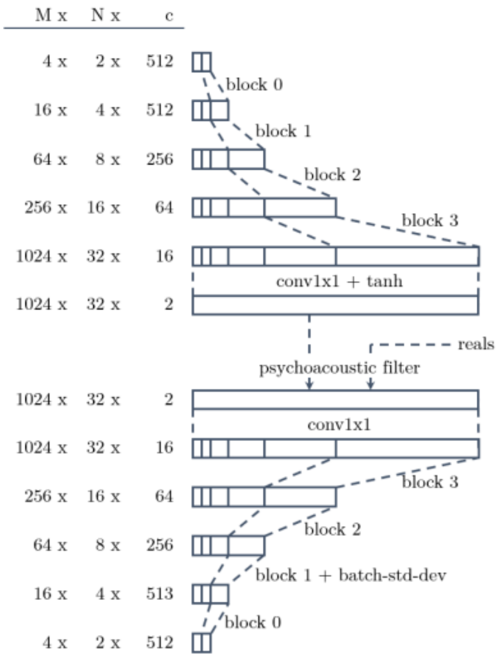

Fig. 5. Convolutional GAN Model Architecture

- The architecture is a deep 2D convolutional network designed to progressively increase the resolution along the time axis, adding higher octaves along the frequency axis. This structure supports the generation of samples with long-range coherence.
- **Number of Layers:**
  - The model includes multiple layers organized in a progression that scales up in complexity.
- **Layers:**
  - **Convolutional layers:** Used in both the generator and discriminator for processing the MDCT amplitude representations.
  - Psychoacoustic layers implement psychoacoustic noise addition to stabilize the GAN training by emulating MP3 quantization noise.
- **Input & Output Dimensions:**
  - The network processes inputs as MDCT amplitudes and outputs stereo audio tracks of 95 seconds length at a 22 kHz sample rate.
- **Loss Functions:**
  - Employs a GAN loss with additional regularization and stability measures, such as adding psychoacoustic noise.
- **Optimizers:**
  - Utilizes the Adam optimizer with specific parameters adjusted for optimal training performance.
- **Compute Resources & Training Duration: :**
  - Training is conducted on a Cloud TPUv2.
  - The model is trained over 250 hours, leveraging efficient computation to reduce the overall training time compared to other high-fidelity audio generation models.

- **Final Output:**
  - The system outputs stereo audio tracks that are coherent and of high quality.
  - Audio outputs are typically stored in uncompressed digital formats like WAV, reflecting the model's ability to generate directly consumable audio content.
- **Evaluation Metrics:**
  - **Audio Quality and Coherence:** The evaluation focuses on the musicality, timbre, and coherence of the generated audio samples, assessing how well the model reproduces realistic and structurally sound music.

This methodology highlights MP3net's innovative integration of convolutional neural network architectures with psychoacoustic modeling to improve the efficiency and quality of music generation. This approach not only reduces the computational demands typically associated with generating raw audio but also enhances the output's realism and coherence.

3 AN APPROACH TO BE REPLICATED!

Based on Compound Word Transformer: ***Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs.*** [Hsiao et al. 2021]

The paper introduces a novel approach to generating full-song music using a Transformer-based model called the Compound Word Transformer (CP). This paper focuses on the generation of expressive Pop piano music and highlights the efficiency of the CP Transformer in producing music of comparable quality to existing models but with significantly faster training and inference times. The key innovation lies in the use of different feed-forward heads in the Transformer decoder to model tokens of various types, such as pitch, duration, velocity, position, tempo, and chord. By grouping tokens into "compound words" and employing a multi-head output module, the CP Transformer offers a unique perspective on music generation.

3.1 Dataset

The dataset used consists of tokens categorized into six types: note-related types [pitch], [duration], [velocity], and metric-related types [position/bar], [tempo], [chord]. For conditional generation tasks, an additional [track] token is employed to differentiate between lead sheet and piano tracks. In the unconditional generation task, the dataset includes tokens designed for generating piano performances from scratch. The vocabulary comprises special tokens such as [ignore], [family=track], [family=note], [family=metric], and [conti]. Various sampling policies are implemented for different token types to ensure diversity in the generated music.

The dataset is structured to represent Pop piano music, incorporating expressive variations in velocity and tempo, with the objective of comparing the proposed CP Transformer with existing models like REMI+XL. The model is trained on a subset of songs, with token embedding sizes customized to the vocabulary size of each token type. The dataset facilitates the evaluation of music generation models based on criteria such as fidelity, richness, humanness, correctness, structure, and overall quality, as evaluated by human subjects.

3.2 Methodology & Intuition

- **Pre Processing** The data preprocessing involves several steps to convert audio files of piano performances into symbolic sequences as follows
  - **Audio to MIDI Transcription:** The audio files are transcribed into MIDI files using the "Onset and Frames" model from Google Magenta. This step provides the pitch, onset time, and velocity of the musical notes from the audio.

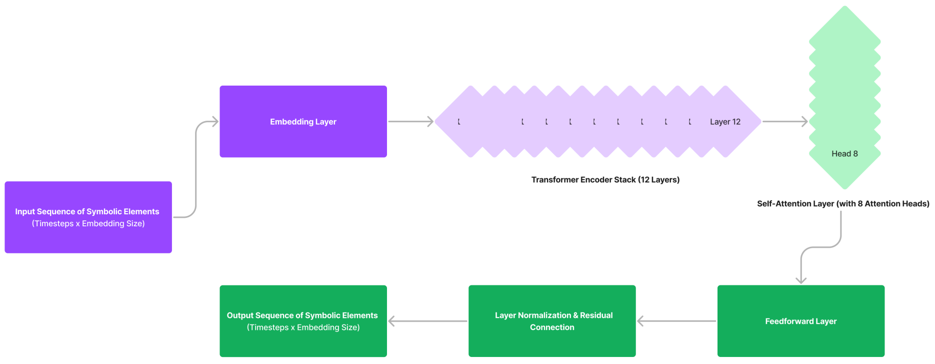
Input Sequence of Symbolic Elements (Timesteps x Embedding Size)
Embedding Layer
Layer 12
Transformer Encoder Stack (12 Layers)
Head 8
Self-Attention Layer (with 8 Attention Heads)
Feedforward Layer
Layer Normalization & Residual Connection
Output Sequence of Symbolic Elements (Timesteps x Embedding Size)

Fig. 6. Compound Word Transformer Model Architecture

  - **MIDI Synchronization:** The MIDI files are then synchronized using the "madmom" library to estimate the downbeat and beat positions. The absolute time is mapped to its corresponding tick, and tempo changes are inferred from the time interval between adjacent beats. This step ensures accurate and consistent timing.
  - **MIDI Analysis:** The synchronized MIDI files are analyzed using an in-house rule-based symbolic melody extraction and chord recognition algorithm. This step extracts the melody notes and chord symbols from the transcription result to form the lead sheets.
  - **Quantization:** The tempo, velocity, duration, and beat positions are quantized to reduce the size of the vocabulary. For example, the 16th note is set as the basic time unit.
  - **Corpus Generation:** The analyzed data is then processed into a corpus format. This involves appending an end-of-sequence (EOS) token to each sequence and quantizing all timing information.
  - **Representation Generation:** The corpus is converted into the desired representation format, which includes the Compound Word (CP) and REMI representations. These representations are used for both unconditional and conditional generation tasks, resulting in four combinations. Each combination involves running specific scripts to generate human-readable tokens, build dictionaries, renumber tokens, and discard disqualified songs that exceed length limits.
- **Model Architecture** The model architecture used in the paper involves a Transformer-based approach with specific characteristics.
  - **Number of Layers:** The model comprises a stack of self-attention layers and feed-forward layers. Specifically, it includes 12 self-attention layers, each with 8 attention heads for fair comparison across models.
  - **Types of Layers:** The self-attention layers operate on a fixed-length sub-sequence of symbolic elements to learn dependencies among the elements. The feed-forward layers process the output of the self-attention layers to make predictions.
  - **Input & Output Dimensions:** The model takes a time-ordered sequence of symbolic elements as input and generates a new sequence of symbolic elements as output. The input sequence is converted into a compressed representation for processing within the model.

  - **Loss Functions:** The model uses negative log-likelihood (NLL) as the loss function during training to measure the discrepancy between the predicted sequence and the ground truth sequence.
  - **Optimizers:** The model employs optimization techniques to update the model parameters during training. Specific optimizers are not explicitly mentioned in the provided text, but common optimizers like Adam or SGD are typically used in training deep learning models.

  The model architecture is designed to learn to generate new sequences of symbolic elements representing music pieces based on the input sequences. It leverages the Transformer's ability to capture dependencies in the data and generate coherent sequences.

- **Compute Used and the Time to Train**
  The paper reports that the model was trained on a single GPU with 11 GB memory. The training time for the model is not explicitly mentioned in the provided text, but it is stated that the model converges 5 to 10 times faster during training compared to state-of-the-art models. This implies that the training time for the proposed model is significantly shorter than that of existing models, allowing for faster development and testing of new music generation models.

- **Evaluation Metrics**
  The evaluation metrics used are detailed and involve both objective and subjective measures to assess the performance of the music generation models. Here is an explanation of the evaluation metrics:
  - **Objective Metrics:**
    * **Melody Matchness:** This metric evaluates the similarity between the lead sheet and the generated piano music in terms of melody. It computes the bar-wise longest common sub-sequence (LCS) of the two sequences and calculates the ratio of matched notes to the total number of melody notes in each bar.
    * **Chord Matchness:** This metric assesses the harmonic similarity between the chord labels in the lead sheet and the corresponding generated piano music. It calculates the segment-wise cosine similarity between the chroma vectors representing each chord label and the generated piano segment.
  - **Subjective Metrics:**
    * **Fidelity:** Measures how faithful the generated music is to the input lead sheet in terms of melody and chord progression.
    * **Richness:** Evaluates the complexity and diversity of the generated music.
    * Humanness: Assesses how natural and human-like the generated music sounds.
    * **Correctness:** Determines the accuracy of the generated music in following the musical rules and structure.
    * **Structureness:** Evaluates the coherence and structure of the generated music.
    * **Overall Quality:** Provides an overall assessment of the generated music based on a combination of the above metrics.

3.3 Our Efforts in Replication

Dataset Preparation

To begin with, the dataset collection involved curating a diverse corpus of music samples encompassing approximately 600 tracks across multiple genres, including sufi, Indian classical, semiclassical, carnatic, bollycap, and ghazals. These tracks were sourced in various formats, such as MP3 and WAV files, necessitating preprocessing to transform them into a suitable format for symbolic representation extraction.

The preprocessing pipeline commenced with audio-to-MIDI transcription, a critical step achieved using advanced tools like the "Onset and Frames" model from Google Magenta. This process extracted key musical elements from the audio tracks, including pitch, onset time, and velocity, thereby converting the audio signals into symbolic MIDI representations.

Following transcription, MIDI synchronization was employed to align the extracted MIDI data with beat positions and infer tempo changes. This synchronization step was crucial for accurately representing timing information, ensuring the fidelity of the symbolic sequences derived from the original audio tracks.

Subsequently, a custom MIDI analysis algorithm was developed to further process the synchronized MIDI files. This algorithm focused on symbolic melody extraction and chord recognition, extracting melody notes, and identifying chord symbols from the transcribed MIDI data.

Further refinement was achieved through quantization, where tempo, velocity, duration, and beat positions were standardized to a common vocabulary size, typically using a 16th note resolution. This quantization step reduced the complexity of the data while maintaining its musical integrity.

Finally, the preprocessed data was structured into a corpus format suitable for model training. This involved appending end-of-sequence (EOS) tokens to signify the conclusion of musical sequences and quantizing all timing information to prepare the dataset for input into the music generation model.

Model Implementation

The model architecture replicated the sophisticated Transformer-based approach described in the research paper. This architecture comprised a stack of self-attention layers and feed-forward layers, facilitating the learning of dependencies among symbolic music elements.

Specifically, the model configuration included 12 self-attention layers, each equipped with 8 attention heads, designed to capture intricate relationships within the input symbolic sequences. This architecture was chosen for its effectiveness in handling sequential data and generating coherent music sequences.

During training, the model employed negative log-likelihood (NLL) as the primary loss function, measuring the discrepancy between the model's predicted symbolic sequences and the ground truth representations derived from the preprocessed music data.

Training & Testing

For efficient data management during model training and testing, the preprocessed dataset was stored as .npz files, ensuring streamlined access and reproducibility of experimental results. Additionally, embeddings representing symbolic music elements were stored in a dictionary format (.pkl), providing a structured means of managing and accessing these representations during model training and inference.

Despite meticulous implementation, the model encountered challenges related to the vanishing gradients problem during training. This issue arose because the loss value reduced below a critical threshold of 0.0029, impeding the model's ability to effectively learn from the training data and generate novel music sequences.

The evaluation phase revealed an average loss of 0.1260 during testing, highlighting potential issues with model generalization and performance. These results underscore the importance of diagnosing and addressing the vanishing gradients problem to enhance the stability and effectiveness of the music generation model.

Challenges Faced

- **Dataset Acquisition and Preprocessing** The initial challenge arose from sourcing and preparing a diverse dataset of approximately 600 music samples in varied formats (MP3 and WAV) covering multiple genres. This process required meticulous preprocessing steps to convert audio files into symbolic representations suitable for model input.
- **Technical Issues and Dependencies** Implementation was impeded by technical challenges, including dependency conflicts and file format inconsistencies encountered during code development. These issues necessitated troubleshooting and adaptation to ensure compatibility and stability in the implementation environment.
- **Model Training and Vanishing Gradients** The most significant challenge surfaced during model training, specifically related to the occurrence of vanishing gradients. This issue was attributed to the model's loss value reducing below a critical threshold, hindering effective learning and the generation of new music samples.

Adaptations & Solutions

- **Custom Dataset Creation** To overcome data availability constraints, a custom dataset spanning diverse music genres was meticulously curated and prepared using a series of preprocessing steps inspired by the techniques outlined in the research paper.
- **Technical Troubleshooting** Adaptations were made to address technical challenges, including resolving dependency conflicts and standardizing file formats, ensuring seamless integration of data processing pipelines within the implementation framework.
- **Model Optimization and Training Strategies** Efforts were directed towards optimizing the model training process to mitigate the vanishing gradients problem. This involved experimenting with learning rate adjustments, implementing gradient clipping techniques, and potentially revising the model architecture to enhance training stability and gradient flow.

In conclusion, this implementation effort represents a thorough adaptation of a sophisticated music generation model to a custom dataset spanning diverse musical genres. Challenges encountered included dataset acquisition, technical integration, and model training, which required adaptive strategies to optimize system performance. While the discussed paper does not directly address Indian music generation, its core concepts—such as type-specific prediction heads and sequence compression via compound words—hold potential for adaptation to other music domains, including Indian music.

4 PROPOSED NOVELTY

4.1 Novelty Claim 1 - The UlTranNet Architecture

The UlTranNet as per our understanding, is a novel architecture, based on its unique integration of wavenet (convolutional layers, dilated convolutions, residual blocks, and skip connections) and transformer-based attention mechanisms specifically tailored for audio processing tasks. This architecture of ours aims to leverage the strengths of both convolutional neural networks (CNNs) and transformers to handle the temporal and spectral complexities inherent in audio signals.

The following features make it unique:

(1) **Hybrid Approach:** This approach of ours combines convolutional layers, which are effective at capturing local and hierarchical features in data, with transformer layers, which excel at capturing long-range dependencies. This hybrid approach is relatively less explored in the domain of audio processing, where either CNNs or RNNs (Recurrent Neural Networks) have traditionally been more common.
(2) **Wavenet Architectural Base - Dilated Convolutions with Residual and Skip Connections:** The model employs dilated convolutions that allow the network to have a larger

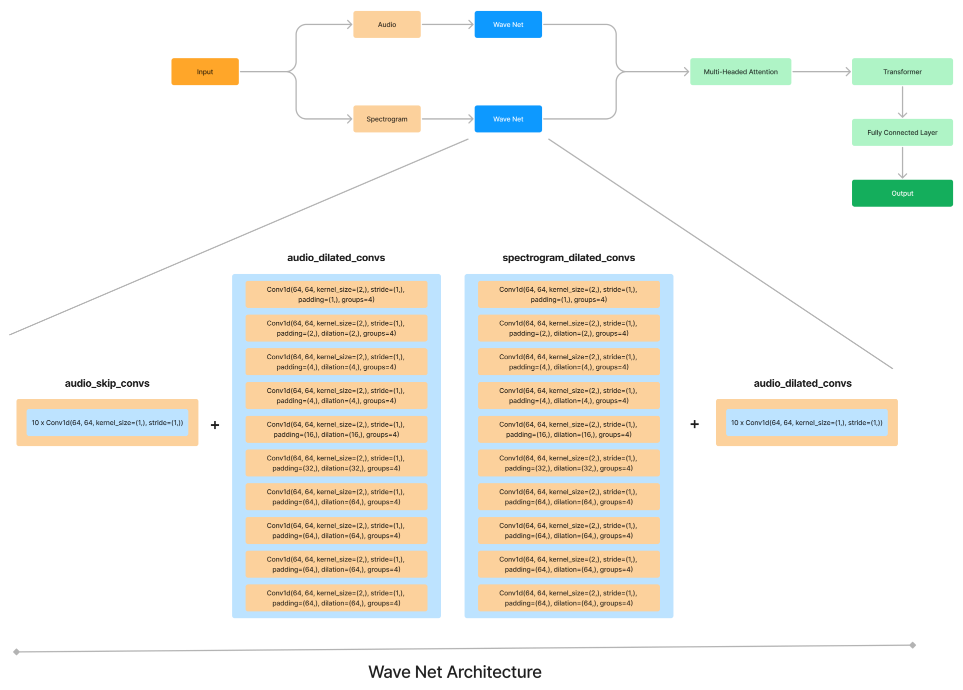
Input
Audio
Spectrogram
Wave Net
Wave Net
Multi-Headed Attention
Transformer
Fully Connected Layer
Output
audio_skip_convs
10 x Conv1d(64, 64, kernel_size=(1,), stride=(1,))
+
audio_dilated_convs
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(1,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(2,), dilation=(2,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(4,), dilation=(4,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(4,), dilation=(4,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(16,), dilation=(16,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(32,), dilation=(32,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(64,), dilation=(64,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(64,), dilation=(64,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(64,), dilation=(64,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(64,), dilation=(64,), groups=4)
spectrogram_dilated_convs
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(1,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(2,), dilation=(2,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(4,), dilation=(4,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(4,), dilation=(4,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(16,), dilation=(16,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(32,), dilation=(32,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(64,), dilation=(64,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(64,), dilation=(64,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(64,), dilation=(64,), groups=4)
Conv1d(64, 64, kernel_size=(2,), stride=(1,), padding=(64,), dilation=(64,), groups=4)
+
audio_dilated_convs
10 x Conv1d(64, 64, kernel_size=(1,), stride=(1,))
Wave Net Architecture

Fig. 7. The UlTranNet Architecture

receptive field, enabling it to capture broader audio contexts without increasing computational complexity significantly. This feature is crucial for handling the temporal dynamics of audio signals. The use of residual and skip connections helps in mitigating the vanishing gradient problem and promotes feature reusability across the network layers.

(3) **Multi-head Attention for Feature Fusion:** The model uses a multi-head attention mechanism to integrate features extracted from separate convolutional pathways for raw audio and its corresponding spectrogram. This attention mechanism allows the model to focus on important features from both input modalities, enhancing its ability to model complex relationships in the data.
(4) **Transformer Encoder for Sequential Processing:** After initial feature extraction and fusion, the model processes the combined features through a transformer encoder. This component is designed to handle sequential data and is adept at managing dependencies across different time scales, which is essential for audio.

The intuition behind this architecture stems from the need to effectively capture both the local features and global dependencies within audio data:

- **Local Feature Extraction:** Audio signals contain rich local structures (e.g., short-term spectral patterns) that are crucial for our task of music analysis. Convolutional layers are well-suited for extracting such features due to their ability to learn spatial hierarchies in data.
- **Global Context Awareness:** Audio processing also requires understanding long-range temporal dependencies (e.g., the complex emotions and their play in a musical piece), which

traditional CNNs are not as equipped to handle. Transformers, particularly with their self-attention mechanisms, provide a means to capture these global dependencies without being constrained by the sequence length.
- **Efficiency and Scalability:** The use of dilated convolutions expands the receptive field exponentially with depth, allowing the network to grasp larger contexts without a linear increase in computational requirements. This makes the model both efficient and scalable, suitable for handling large-scale audio datasets.
- **Robust Feature Integration:** By employing a multi-head attention mechanism to integrate audio and spectrogram features, the model ensures that it does not overlook the complementary information provided by different representations of the audio. This robust integration is key to improving the model's performance on complex audio processing tasks.

4.1.1 ***Supporting Experiment:*** The experiment involves training the UlTranNet model on a dataset of raw audio files of the format .mp3 and .wav, with their standard sampling rate (44KHz) and standard metadata. The dataset is processed to extract both raw audio and its corresponding spectrogram representations, which serve as inputs to the model. This experiment is structured to evaluate the model's performance in generating or transforming audio based on these inputs.

Data Pre-processing

We utilize a custom dataset that consists of audio files collected and/or pre-generated within a specified directory. These audio files are of the format '.mp3' or '.wav', which are common formats for storing audio data. Our predefined class handles the loading and preprocessing (down-sampling, padding and trimming) of these files into a format suitable for input into the neural network:

Each audio file is loaded and resampled to a consistent sample rate of 22050 Hz and the duration is also standardized to a default of 5 seconds, with padding or truncation applied as necessary. This was done to ensure our training completion of the model in the given time frame with our compute resources. A spectrogram is generated for each audio sample using the Short-Time Fourier Transform (STFT). This transformation converts the time-domain audio signals into a frequency-domain representation, which is crucial for many audio processing tasks.

The dataset, post its pre-processing is divided into three subsets: training, validation, and testing. The splits are defined as follows:

- **Training Data (70%):** Used to train the model. This data helps the model learn to generate or transform audio by adjusting its weights based on the defined loss functions.
- **Validation Data (15%):** Used to tune the hyperparameters and prevent overfitting. This subset helps in validating the model's performance during training, allowing for adjustments before final evaluation.
- **Test Data (15%):** Used to assess the model's performance after training. This data is crucial for evaluating how well the model generalizes to new, unseen data.

Training & Evaluation

The model is trained using a custom composite loss function that combines (simple addition for now) perceptual loss, spectrogram loss, and feature matching loss. These loss functions are designed to ensure that the generated audio is similar to the target audio in terms of perceptual quality, spectral content, and discriminative features.

- **Perceptual Loss:** Measures the difference in high-level features extracted by a pre-trained model (VGGish).
- **Spectrogram Loss:** Measures the difference (L1 Norm) in the spectrogram representations of the generated and target audio.

- **Feature Matching Loss:** Measures the similarity (L1 Norm) in features extracted by an intermediate layer of a discriminator model (The input dimensions and the output dimensions are to be carefully chosen for this particular loss implementation).

The training process involves feeding batches of audio (A batch size of 10 in our case) and their corresponding spectrograms to the model, calculating the composite loss and updating the model's weights using backpropagation. The performance is periodically evaluated on the validation set during training to monitor progress and adjust learning parameters if necessary.

Modifications and Innovations

The primary innovation in this experiment is the integration of the transformer architecture with the wavenet architecture especially for audio processing and feature extraction, along with the hybrid recombination of the audio and the spectrogram data with the Multi-Head Attention. This approach of ours aims to leverage the strengths of these architectures to improve the model's ability to handle complex audio processing tasks. Additionally, the use of a composite loss function is a strategic choice designed to optimize the model's performance across multiple dimensions of audio quality.

In toto, we hope this experiment's setup, including the data handling, model architecture, and training strategy, represents a comprehensive approach to equate, and hopefully advance the state of the art in audio processing for music generation using deep learning techniques.

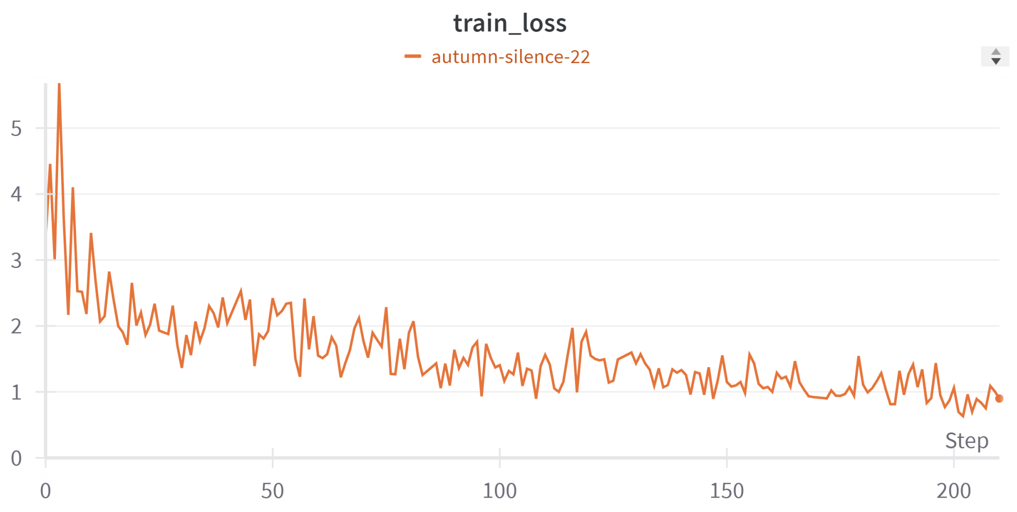

Fig. 8. UlTranNet Train Loss Plot - 4.32 Epochs

4.1.2 ***Result:*** Regrettably, due to time constraints, the generation of music output files in '.wav' or '.mp3' formats was not feasible within the allotted timeframe. However, our model's training process exhibited a remarkable trajectory, providing valuable insights into its learning dynamics and potential capabilities.

Utilizing a compute power of 40 GB GPU RAM over a total training time of 4 hours, our model successfully completed 4 epochs and 39 batches. Throughout this training period, we observed a

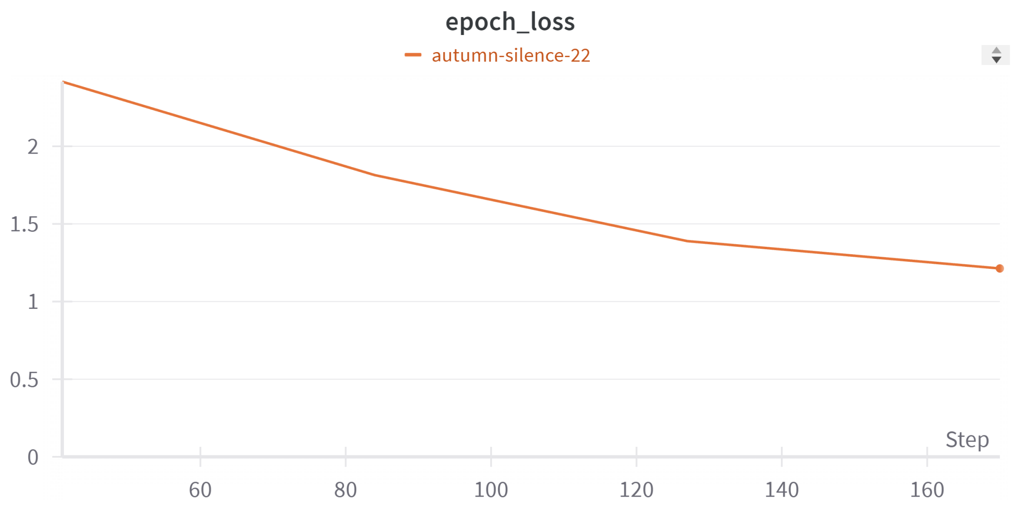

Fig. 9. UlTranNet Train Epoch Plot - 4 Epochs

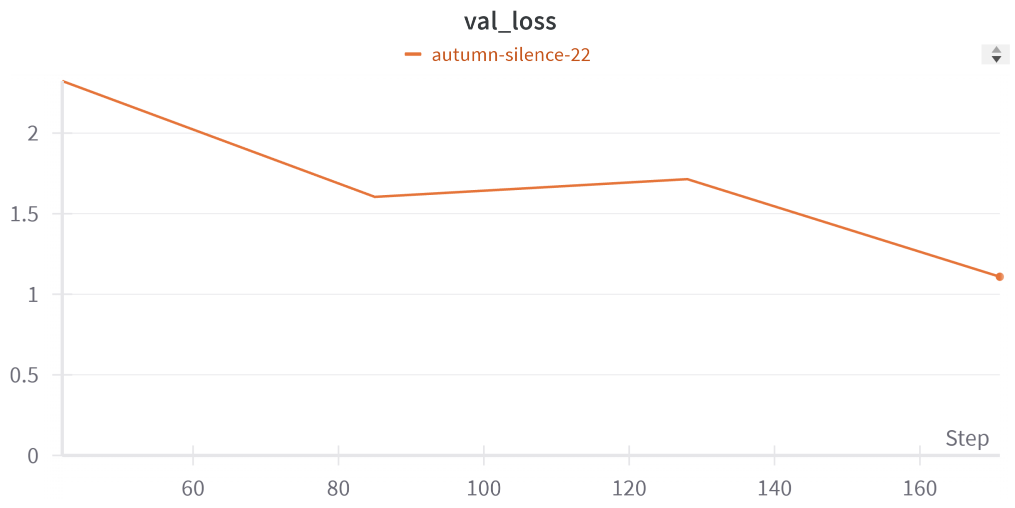

Fig. 10. UlTranNet Validation Loss Plot - 4 Epochs

gradual decrease in both epoch and validation loss values, indicative of the model's ability to learn and generalize from the training data.

Specifically, the epoch loss decreased from an initial range of [5.899, 0.632], while the validation loss showed a similar decreasing trend from [2.323, 1.109]. Moreover, the average epoch loss exhibited a consistent downward trajectory, mellowing down from [2.416, 1.213].

Further analysis of the model's learning dynamics revealed well-behaved gradients, as evidenced by observations from the saved checkpoint files. This suggests that the model's optimization process is proceeding smoothly, with gradients properly aligned for effective learning.

Extrapolating from these observations, we posit that with sufficient compute power and time, our model demonstrates promising potential to efficiently learn the intricate nuances and relationships between the spatial and temporal aspects of music. This extends to the realm of Indian classical music, where the model could potentially capture the complex structures of Raga (melodic framework) and Tala (rhythmic cycle), further enriching its ability to generalize from a diverse dataset encompassing varied musical samples.

4.2 Illustrations and Lessons Learned

To illustrate, currently, we possess only the audio sequences represented as torch tensors. Due to limitations in compute power, we are unable to load the model and execute it locally. To overcome this constraint, we aim to acquire additional compute resources and time. Our plan involves converting these torch tensors into '.wav' audio outputs along with their corresponding spectrograms. This process will enable us to visually inspect and correlate the generated audio with the original source, providing valuable insights into the model's performance.

Lessons learned from this endeavor encompass several key aspects. First and foremost, meticulous attention to dimensionality is paramount when designing and implementing a neural network architecture, especially when building from scratch. Furthermore, adopting optimized and efficient implementation methodologies enhances the flexibility and utility of the model, allowing for smoother integration into various applications.

Diversifying the dataset emerges as a prudent decision for fostering a generalized framework capable of handling diverse scenarios and inputs effectively. Moreover, the significance of acquiring a substantial dataset containing domain-specific data cannot be overstated in the realm of deep learning. A well-curated dataset serves as the cornerstone for training robust and accurate models, provided the architecture is well-structured.

Beyond the technical realm, this project has also imparted valuable lessons in time management, team coordination, resource allocation, workflow optimization, and other essential skills. These soft skills play a pivotal role in the success and efficiency of any research endeavor, complementing the technical expertise and contributing to overall project success.

4.3 Novelty Claim 2 - MFCCs as a feature representation for music generation

This idea does not represent a completely novel concept in the field of audio processing and synthesis as the use of MFCCs (Mel Frequency Cepstral Coefficients) as input features for neural networks, particularly in tasks related to audio such as speech recognition, music classification, and even music generation, has been explored in various studies and applications. However, we hope that the specific implementation details and the combination of techniques can contribute to a unique application or improvement over existing methods.

In this claim, we suggest experimenting with MFCCs as a feature representation for generating music using a variant of WaveNet. MFCCs are a representation of the short-term power spectrum of sound, derived by taking the log of the pow - at various frequencies along the Mel scale. These are widely used in audio processing because they effectively capture timbral and textural aspects of sound, making them highly informative for audio analysis tasks.

The intuition behind using MFCCs for music generation in our approach stems from several key properties of MFCCs like **Human Perception Alignment** - MFCCs are designed to mimic the human auditory system's response, making them particularly effective for tasks where human-like processing of sound is beneficial, **Dimensionality Reduction** - MFCCs reduce the dimensionality of audio data (compared to raw audio waveforms), which can simplify the learning process for neural networks, and **Effective Feature Representation** - They encapsulate important characteristics of sound such as timbre, which are crucial for distinguishing different types of sounds and musical tones.

By using MFCCs as input to the subtly modified WaveNet, which is capable of generating high-quality audio, the hope is to leverage the efficient representation of MFCCs to guide the network in generating musically coherent and pleasing sounds. The network would ideally learn to map the space of MFCC features back to the audio domain, potentially capturing intricate nuances of music that are encoded in these features.

While using MFCCs for audio-related tasks is not novel in itself, the specific application to music generation using a deep generative model like WaveNet along with MFCCs hopefully brings new insights or improvements.

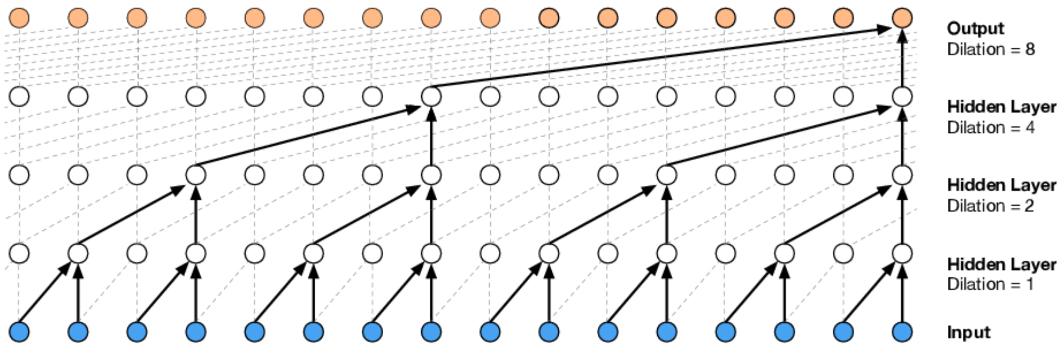

Fig. 11. The Wavenet Model

4.3.1 *Supporting Experiment:* Training Data

The training data consists of audio files, specifically in MP3 format, which are preprocessed to extract Mel Frequency Cepstral Coefficients (MFCCs). These MFCCs serve as the input features of dimensions (2,32,9923) for the neural network, with a target dimension of (32, 9923). The dataset is split into training, validation, and test sets with proportions of 70%, 15%, and 15% respectively, just as in the previous case. This division ensures that the model is trained on a majority of the data while still having separate sets for tuning and evaluating performance.

Test Data and Metrics

The test set, like the training set, consists of MFCCs extracted from MP3 audio files not seen by the model during training. The performance on the test set is evaluated using the Mean Squared Error (MSE) between the model's audio output and the true audio waveform. We obtained an average MSE of ***0.0405*** on the Test Set, indicating the average squared difference per sample between the predicted and actual audio waveforms. This metric helps in quantifying the accuracy of the audio generated by the model in comparison to the original audio.

Losses and Optimizers

The model utilizes the Mean Squared Error (MSE) loss function, which is appropriate for regression tasks such as audio waveform generation where the goal is to minimize the error between the predicted and actual values. The optimizer used is Adam, a popular choice for deep learning applications due to its efficiency in handling sparse gradients and its adaptive learning rate capabilities, which help in converging to the optimal solution more effectively.

Methodology

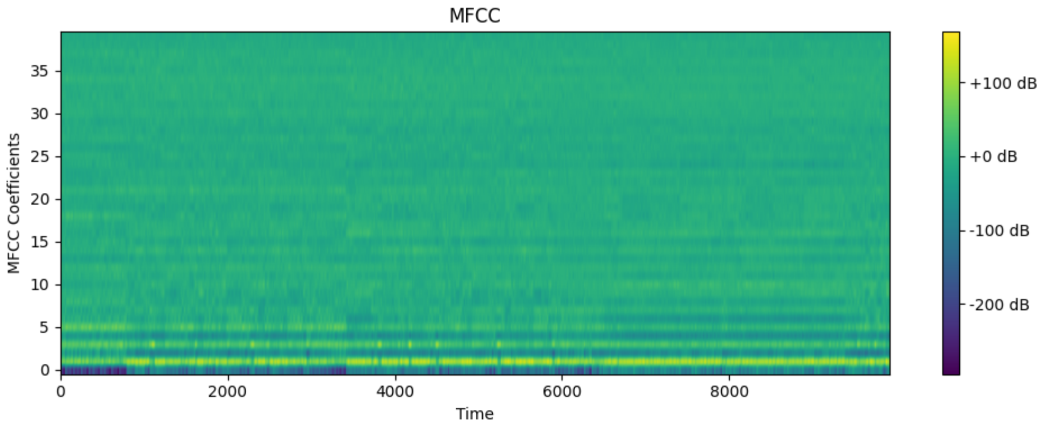

Fig. 12. The MFCC Plot of a Randomly picked sample from the Train Dataset

The core of the methodology involves using a WaveNet-like architecture (We modified a few cells internally and added a few residual blocks unconventionally), known for its ability to produce high-quality audio. This model is adapted to take MFCCs as input instead of raw audio waveforms, leveraging the compact and informative representation of audio provided by MFCCs. To subtly emphasize, the model architecture includes dilated causal convolutions that allow it to capture long-range dependencies in audio data, crucial for generating coherent musical pieces.

The training process involves feeding batches of MFCCs (of size 40 in our case) into the model, which then attempts to reconstruct these MFCC bands. The output from the model is the MFCC band, which is compared to the actual MFCC band to compute the loss, which is then used to update the model's weights through backpropagation. Finally, these MFCC bands are converted into '.wav' outputs by the inverse MFCC mapping, thus generating the audio file.

Modifications and Innovations

This approach hypothesizes that MFCCs can provide a more computationally efficient and potentially more effective way of generating music due to their reduced dimensionality and their focus on perceptually relevant features of sound.

Additionally, the inverse transformation process — converting MFCCs back to audio — is carefully handled to ensure that the generated audio maintains quality and coherence. This involves techniques to approximate the inverse of the MFCC extraction process, which is non-trivial and crucial for the success of the experiment.

Overall, This experiment explores the application of MFCCs with a modified WaveNet architecture for music generation, focusing on computational efficiency and the preservation of musical quality. The results, indicated by an Average MSE of 0.0405 on the test set, suggest that the model can effectively generate audio from MFCCs, although further improvements and evaluations are necessary to fully validate the approach.

4.3.2 Results. During the training phase, the model's performance was evaluated by computing both epoch and validation losses. The epoch losses ranged between 0.03 and 0.04, exhibiting either a fluctuating behaviour at a learning rate of 0.1. or a consistent behaviour throughout the training process at a learning rate of 0.01. Similarly, validation losses remained within the range of [0.02, 0.04], demonstrating stable convergence. Clearly, employing a learning rate of 0.01 yielded satisfactory results, while a higher learning rate of 0.01 led to multiple fluctuations and chaos.

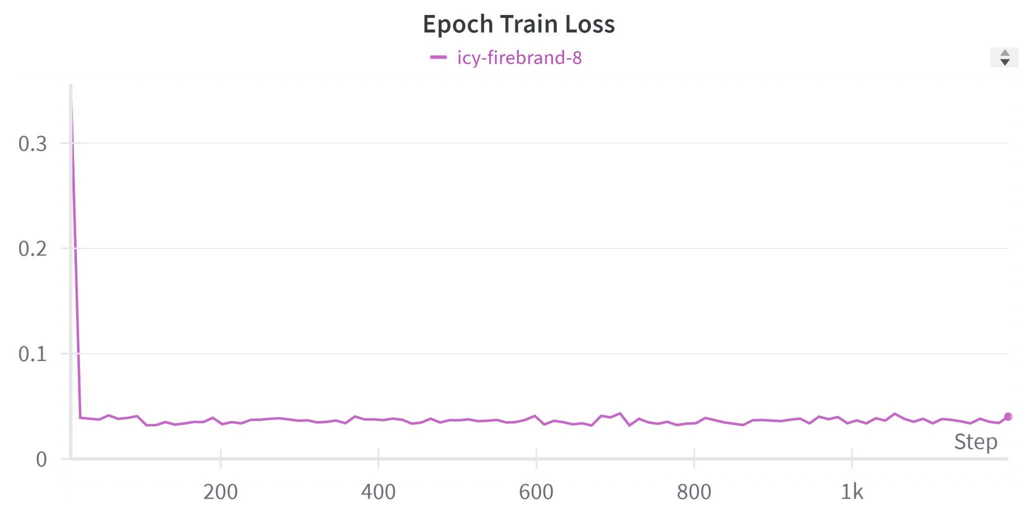

Fig. 13. Modified Wavenet Epoch Loss - 200 Epochs, Lr = 0.01

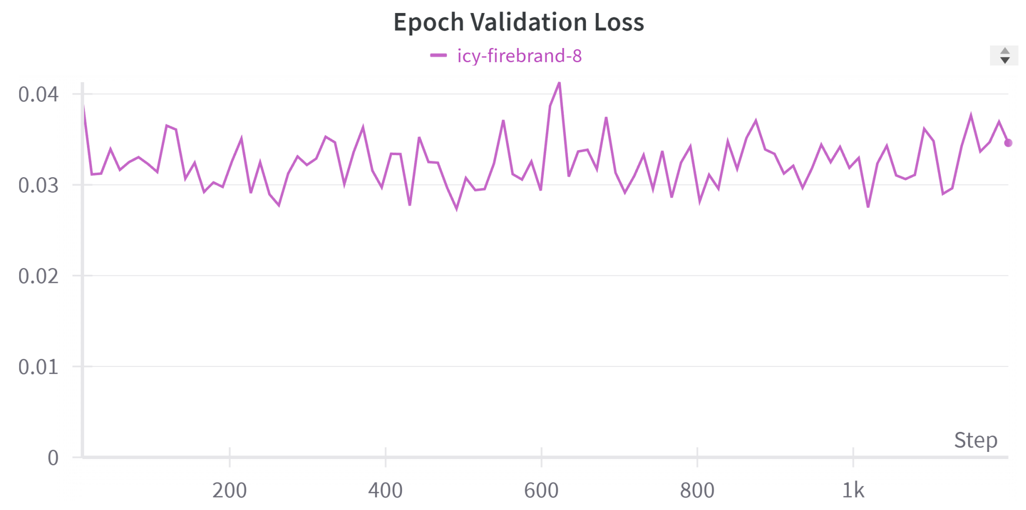

Fig. 14. Modified Wavenet Validation Loss - 200 Epochs, Lr = 0.01

Upon testing the trained model on the unseen test dataset, the Mean Squared Error (MSE) was measured at 0.0405. This value suggests either a close correspondence between the generated output and the original MFCC band of the test set, or indicates potential areas for further model refinement and learning improvement, given that the model couldn't train well at all.

It's essential to note that this experiment was conducted primarily for exploratory purposes, and thus, did not aim to yield definitive or conclusive outcomes. Instead, it served as a foundational

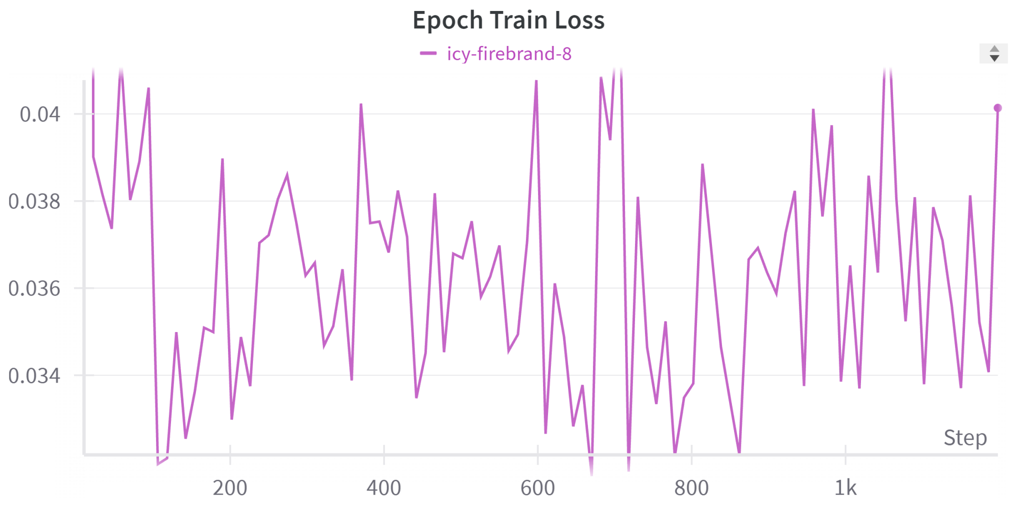

Fig. 15. Modified Wavenet Epoch Loss - 200 Epochs, Lr = 0.1

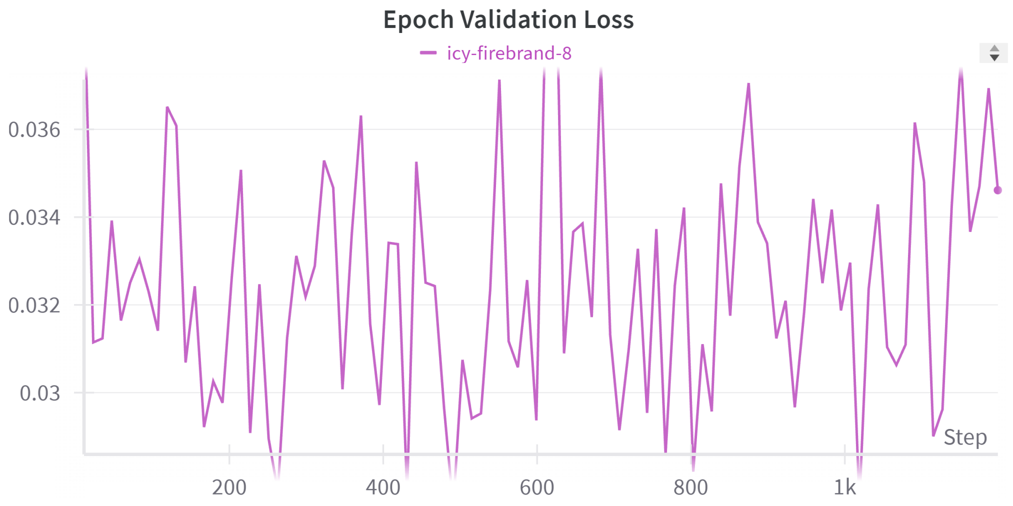

Fig. 16. Modified Wavenet Validation Loss - 200 Epochs, Lr = 0.1

step towards understanding the model's capabilities and potential avenues for future research and optimization.

4.3.3 Illustrations and Lessons Learnt. In our exploration of using MFCCs with a modified WaveNet architecture, several notable improvements and areas for refinement emerged, shedding light on the efficacy and challenges of this approach.

Improved Aspects:

- **Computational Efficiency:** Utilizing MFCCs as input features instead of raw audio waveforms significantly reduced the computational complexity of the model. By focusing on the perceptually relevant features of sound, the model could generate high-quality audio with fewer computational resources, making it more scalable for real-world applications.
- **Long-Range Dependencies:** The incorporation of dilated causal convolutions in the WaveNet-like architecture facilitated the capture of long-range dependencies in audio data. This allowed the model to generate musical pieces with smooth transitions and nuanced variations, enhancing the overall coherence and realism of the generated audio.

Areas for Refinement:

- **Model Training Stability:** While the model demonstrated stable convergence during training, fluctuations in epoch losses were observed, particularly when using a higher learning rate of 0.1. Further investigation is needed to address these fluctuations and improve the stability of the training process, potentially through adaptive learning rate strategies or regularization techniques.
- **Generalization to Test Data:** The Mean Squared Error (MSE) on the test set was measured at 0.0405, indicating a relatively close correspondence between the generated output and the original MFCC band of the test set. However, the performance varied across different learning rates, suggesting the need for further optimization to enhance generalization to unseen data and mitigate overfitting tendencies.
- **Inverse Transformation Process:** The inverse transformation process, which converts MFCCs back to audio waveforms, requires careful handling to ensure the quality and coherence of the generated audio. Future research should focus on refining this process to improve the fidelity of the generated audio and minimize artefacts introduced during reconstruction.

Irrespective of the results and the outputs, we believe UlTranNet has the potential to be a robust and modular music generation model, we hope to train it and build the entire envisioned model soon. Thank You for the great learning experience.

REFERENCES

Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. 2020. *Deep Learning Techniques for Music Generation.* Springer International Publishing. https://doi.org/10.1007/978-3-319-70163-9

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. 2020. Jukebox: A Generative Model for Music. arXiv:2005.00341 [eess.AS]

Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. 2021. Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs. arXiv:2101.02402 [cs.SD]

Rachel Manzelli, Vijay Thakkar, Ali Siahkamari, and Brian Kulis. 2018. Conditioning Deep Generative Raw Audio Models for Structured Automatic Music. *ArXiv* abs/1806.09905 (2018). https://api.semanticscholar.org/CorpusID:49434445

Korneel van den Broek. 2021. MP3net: coherent, minute-long music generation from raw audio with a simple convolutional GAN. arXiv:2101.04785 [cs.SD]

Ääron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. In *Arxiv.* https://arxiv.org/abs/1609.03499

Ziyu Wang, Dingsu Wang, Yixiao Zhang, and Gus Xia. 2020. Learning Interpretable Representation for Controllable Polyphonic Music Generation. arXiv:2008.07122 [cs.SD]

Richard K. Wolf. 2003. *Asian Music* 34, 2 (2003), 133–139. http://www.jstor.org/stable/4098462

Xianchao Wu, Chengyuan Wang, and Qinying Lei. 2020. Transformer-XL Based Music Generation with Multiple Sequences of Time-valued Notes. arXiv:2007.07244 [cs.SD]