UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN ICT FOR INTERNET AND MULTIMEDIA

"Generative Models for Music Using Transformer Architectures"

Relatore: Prof. Antonio RODA'

Laureanda:Sıla ÖZTÜRK

Correlatore: Filippo CARNOVALINI
Alessandro FIORDELMONDO

ANNO ACCADEMICO 2022– 2023

Data di laurea 19.10.2023

Abstract

This thesis focuses on the growth and impact of Transformers architectures which are mainly used for Natural Language Processing tasks for Audio generation. We think that music, with its notes, chords, and volumes, is a language. You could think of a symbolic representation of music as human language.

A brief sound synthesis history which gives basic for modern AI-generated music models is mentioned. The most recent in AI-generated audio is carefully studied and instances of AI-generated music are told in many contexts. Deep learning models and their applications to real-world issues are some of the key subjects that are covered.

The main areas of interest include transformer-based audio generation, including the training procedure, encoding and decoding techniques, and post-processing stages. Transformers have several key advantages, including long-term consistency and the ability to create minute-long audio compositions.

Numerous studies on the various representations of music have been explained, including how neural networks and deep learning techniques can be used to apply symbolic melodies, musical arrangements, style transfer, and sound production.

This thesis focuses on transformer models and their importance in the context of AI-based generative models, particularly in terms of their emotional influence on individuals. We train transformer models on a dataset of diverse noise sounds to demonstrate their ability to reliably generate sounds in response to textual stimuli. We hope to establish the usefulness of these models in producing the intended sounds by thorough human evaluation. Furthermore, we investigate the emotional responses evoked by these created sounds in order to gain a better understanding of how people perceive and interact with them.

Overall, this thesis not only improves the field of generative models with transformer design for music composition, but it also provides a thorough examination of and the effects of created sounds on individuals. It emphasizes the tremendous possibilities of this technology by showing cutting-edge developments in AI-generated sound synthesis.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, the use of artificial intelligence (AI) has increased significantly in many fields. AI is now a part of our daily lives, making it easier and faster. It's used to solve a variety of everyday problems, from smartphones to customer service. As artists, we can not only solve the difficulties of everyday life but also in the field of visual art. Still, there are some visual problems with AI as an artist, since it's a kind of argument that anything AI produces is considered art.

It is used to achieve beautiful results in a variety of creative fields, from image generation to sound generation to text. Nevertheless, we can debate what is beautiful and whether we can assume that AI can create beautiful things. It's easy to say that AI is very successful in completing tasks in the objective realm, but in the subjective realm, it will take years before we can accept whether the work it produces is artistic or not. It also costs. However, this paper is also about sound creation and is not seeking an artistic aspect. AI methods for music and audio generation are discussed, with a particular focus on their artistic and technical characteristics. Because sound production is mainly used in the artistic field.

The Transformers architecture has seen an increase in usage and application due to the improvements made in the field of AI and Natural Language Processing (NLP). The Transformers approach was first developed for NLP tasks, but it has now been extended to a diverse range of abilities, including audio generation. The focus of this thesis is on a comprehensive analysis of the expansion

and profound impact of this architecture on music and audio synthesis.

Music and natural languages have similar characteristics when it comes to characterizing their properties, such as notes, velocity, chords, and volumes. Patterns in human language are mirrored in the symbolic representation of music. To explore new methods for AI-generated audio, this research utilizes the benefits of these similarities and delves deep into the perspective of neural network architecture. The most recent developments in resolving instances of various AI-composed music are given special attention due to the rich pattern of AI-generated audio.

The main part of this research is the wide observation of deep learning models in the sound generation area, defining its difficulties and finding the best match for the objective of the subject. While focusing on transformers base generation, navigating training process challenges, and deeply understanding encoders and decoders methods. Transformers have benefits such as generating long-term and consistency. Those advantages of transformer-based models are investigated both from waveform and symbolic representation.

While this research investigates generation strategies, several types of music representation are also highlighted and their obstacles are studied. There are various approaches in neural networks and deep learning for symbolic models to produce sounds, such as musical arrangement and style transfer, to reduce difficulties. The major topic of this thesis is explaining transformer models with the use of GAN models. As a result, the significance and impact of GANs cannot be overstated.

In summary, the goal of this thesis is to broaden the understanding of transformer-based generative models and to contribute to the field of generative models. Not only does it highlight the most recent research on transformer-based generative models, but it also discusses how to analyze musical data and how to generate it.

1.1 Explanation of Sound Generation with AI

Objects that vibrate, as we know, produce sounds. Typically, humans make certain noises by using an instrument, either a musical instrument or any other instrument, such as a wooden table. Human vision enables individuals to use and be creative with sounds. Electronic music or sound became popular as technology advanced, and not only with various materials, but also with computers, which allow us to produce, change, or blend sounds to create many types of sound, voices, and music.

Nowadays, AI is prominent in almost every industry, including sound and music. AI speech synthesis technology has been improved by incorporating fresh new artificial intelligence technology, allowing for the creation of not only realistic sounds but also performance nuances that truly sound like a human is performing.

Sound has five important properties: wavelength, amplitude, frequency, time period, and velocity. They can be used to generate sound as information.

1.2 Brief history of sound synthesis

Art, like every other area of AI, is experiencing rapid growth. Nowadays, it has significant success in generating creative output, which is a significant challenge for AI. Because AI lacks the eyesight that humans have, but even with limited resources, AI produces results that humans cannot discern.

It can write books, finish stories, paint, photo-shop photographs, and produce music of various genres, among other things. There have been various approaches for making music, such as shifting the dimension of audio data and transporting it to 2-dimensional spectrum space to appear as visuals and processing them as images. The second way is to use a symbolic representation of music and treat it as a language before converting it to an auditory representation.

These various approaches enable AI developers to experiment with various types of deep learning models to process and generate.

1.3 Aims and Objectives

The goal of this thesis is to make sounds from raw wave audio data that can affect neurons in the brain and influence the flavor of foods. Several studies have found that our sensory organs are interconnected and that when one of them is replicated, the others can be affected as well. The effect of color on taste has been studied, and it has been discovered that although lighter colors make us feel sweeter, darker ones make us feel bitter[26]. As a result, not only vision but also hearing sense might influence how we taste food.

When we are eating something, we feel more fresh and crunchy when we hear the voice that comes from the food texture, such as being crunchy. If the frequency and decibel levels are raised, the meal will be perceived as more crunchy. It has since been discovered that some neuro-acoustic sounds that can affect neurons in the brain can also affect the type of taste that can be sensed by the brain. [32].

With the knowledge of music, its waveform and description, such as taste, it will be employed in this thesis, the model may generate similar sounds with the same aim as the sound we received references for.

1.4 State of the Art

This thesis explains the current development status of products based on cutting-edge technology, ideas, and features. It refers to the highest level of overall development of a device, technology, or scientific subject at any given time. The phrase is also known as "cutting edge" or "state of the art" and is used to define the cutting edge of these technologies, as well as their latest, newest, most advanced versions.

Although AI technologies are likely to be considerably younger than many of the technologies that have influenced and sustained this decade, (AI) technology has evolved rapidly over the years. AI technology is continually evolving, and it is vital to understand which technologies are completely new and where these new concepts might be applied. It is also difficult to find acceptable designs that can be tracked quickly, efficiently, and precisely using innovative techniques to

generate either sound, graphics, or text, depending on the purpose. Because there are numerous problems in sound generation, many ways can be combined to produce better outcomes. It is not entirely novel, but it can answer the problem at hand and can be integrated with another design or method to achieve a better solution.

Deep learning models paired with high-quality image data can be utilized to solve real-world challenges like medical image analysis, video conferencing, and autonomous driving. However, efficiently addressing these difficulties with AI is a challenge. In many circumstances, only the best model, also known as cutting-edge, can match the constraints of a certain application.

State-of-the-art (SOTA) DNNs are the finest models for a wide range of tasks. A DNN can be classified as SOTA depending on its accuracy, speed, or any other relevant parameter. In most computer vision fields, however, there is a trade-off between these metrics. So, while a rapid DNN is possible, its precision is limited. The metrics we calculate from errors according to tasks are the criteria that are typically used to compare and assess DNNs. For DNNs, the state-of-the-art is based on a combination of accuracy and error measures, as well as related additional performance metrics.

SOTA DNNs are the best models which can be used for any kind of task. A DNN can be defined as SOTA based on its accuracy, speed, or any other metric of interest. However, in most computer vision fields, there is a trade-off between these measurements. So, you can have a very fast DNN, but its accuracy is not as targeted. The criteria that are usually used to compare and evaluate DNNs are the metrics we calculate from errors according to tasks. State-of-the-art DNNs are based on a combination of accuracy and error metrics and related additional performance metrics.

Examples of AI-generated sounds across different industries

Artificial intelligence (AI) developments and the proliferation of deep learning techniques have ushered in a new era of creativity and invention in a variety of industries in recent years. One of the notable areas where AI has made tremendous breakthroughs is in audio and sound creation. This chapter looks

at a variety of applications and cases where AI-generated sounds have made their way into many industries, altering the landscape of creativity and functionality.

Music and Entertainment Industry

The music and entertainment industries stand to benefit the most from AI-generated sounds. AI has proved its ability to write original pieces of music that resonate with human emotions by fusing machine learning algorithms and musical theory. From classical symphonies to modern rhythms, AI models may detect patterns in enormous music databases and develop songs that mimic the styles of famous composers or give whole new audio experiences. Furthermore, AI-generated sound effects have transformed the post-production process in the film and game industries. AI algorithms contribute to the enhancement of visual narrative by fast building immersive audio landscapes, thereby increasing audience engagement.

Healthcare and Wellness Sector

Beyond entertainment, the healthcare and wellness sector has embraced AI-generated sounds to facilitate therapeutic interventions and improve patient well-being. Sound therapy, an age-old practice rooted in ancient cultures, has evolved with the integration of AI. By leveraging machine learning techniques, AI systems can tailor soundscapes that promote relaxation, stress reduction, and sleep improvement. These personalized auditory environments adapt to an individual's physiological responses, ensuring optimal therapeutic outcomes. Additionally, AI-generated sounds have found utility in diagnosing medical conditions. The analysis of audio patterns, such as respiratory sounds or heart murmurs, assists medical professionals in making accurate assessments, thus accelerating the diagnostic process.

In conclusion, Chapter 2 delves into the multifaceted applications of AI-generated sounds across different industries. From the harmonious compositions of the music and entertainment industry to the therapeutic soundscapes of healthcare, AI's transformative potential in sound generation is undeniable. As we traverse these examples, the subsequent chapters will delve deeper into the technical underpinnings that enable such creative and functional feats in the realm of AI-generated audio.

1.5 Applications of Sound Generation with AI and Deep Learning

1.5.1 Role of AI-generated sounds in industries such as music, film, gaming, and healthcare

Artificial intelligence AI is enabling breakthrough advancements in a variety of industries, altering and redefining old practices. This chapter investigates the critical role that AI-generated sound plays in a variety of disciplines, including music, cinema, games, and healthcare. A thorough examination of these industries reveals how AI-generated music is transforming creativity, user experience, and therapeutic uses.

Reshaping the Music Industry

The music industry has undergone a transformative journey with the integration of AI-generated sounds. AI algorithms, driven by intricate neural networks, can analyze vast troves of musical data, discerning patterns, harmonies, and rhythms. This newfound ability has enabled AI systems to compose original musical pieces, paying homage to established genres or even forging entirely new sonic landscapes. Musicians and composers now collaborate with AI counterparts, leveraging their expertise to create compositions that challenge conventional boundaries. Additionally, AI-generated sounds have empowered artists to swiftly experiment with different melodies, arrangements, and instrumentation, fostering a culture of dynamic creativity and exploration.

Enhancing Cinematic and Gaming Experiences

AI-generated sounds have emerged as vital tools for creating immersive and engaging aural landscapes in film and games. The complex interplay of sights and audio has a significant impact on audience engagement and emotional resonance. AI-powered sound effects and compositions provide these sectors with new levels of personalization and adaptability. Machine learning algorithms assess visual signals and scenarios to generate audio pieces that heighten tension, elicit emotions, and sync flawlessly with the evolving narrative. This collaboration improves the whole cinematic and gaming experience, ensuring that fans

are immersed in a multi-sensory trip that extends beyond the screen.

Innovating Healthcare and Therapeutic Applications

AI-generated sounds have made advances in healthcare, contributing to novel therapeutic techniques and diagnostics. Sound therapy, which has long been acknowledged for its ability to reduce stress and promote relaxation, has been enhanced by AI's analytical powers. Personalized auditory experiences are curated by AI systems based on human preferences and physiological responses. These soundscapes help to reduce anxiety, improve sleep quality, and promote general well-being. Furthermore, AI-powered acoustic pattern analysis provides a speedy and accurate medical diagnosis. AI-generated noises are used by healthcare experts to detect irregularities in heartbeats, respiratory cycles, and other physiological signals, speeding up diagnosis and therapy planning.

In this chapter, we have explored the multifaceted impact of AI-generated sounds across the music, film, gaming, and healthcare industries. By illuminating the profound transformations these sectors have undergone, we lay the foundation for deeper technical investigations in subsequent chapters. The intricate dance between AI algorithms and soundscapes continues to shape the present and future landscape of creativity, entertainment, and wellness.

1.5.2 Potential future developments within the field of AI-generated sounds

There are currently a few examples of possible future results, such as music and voices generated by artificial intelligence software based on sophisticated inputs such as content commands and images.

Musical advancements will evolve to the point that we will be able to use them to produce new and dynamic live music shows in the future. This program enables musicians to rapidly compose and improvise modern melodic compositions, resulting in a one-of-a-kind performance each time.

1.6 Some Transformer Models For Music And Sound Generation

There are several research and examples about audio and music generation according to their purpose. From jazz music generation to pop music. From text-based to wave-based. As a symbolic representation, those can be from piano notes to guitar tabs, and even for percussive instruments, there are many types of research.

1.6.1 AudioCraft

AudioCraft is Facebook research which includes several modules such as EnCodec, MusicGen, and AudioGen. The aim of this project is to generate music or some sort of voice, or acoustical audio from a given text. It includes an NLP task to match the description with the type of music or sounds it is trained with. According to the description, music, or sound given, it started to generate related sounds.

The AudioCraft project allows you to choose a conditioner, language model, and solver according to the aim of the task to be processed to generate sounds. There are four kinds of solvers, AudioGen generates atmosphere, an acoustical environment which is trained by environmental sounds such as bird voice, and instrumental sound. MusicGen is way smaller than AudioGen according to the samples that has to deal with because it just responsible for musical data not every kind of sound data in the environment. EnCodec is the comparison task for both AudioGen and MusicGen which includes an encoder-decoder transformer-based model. The last solver is diffusion which is for the task multi-band-with training.

MusicGen MusicGen is the task that includes a language model specific to music generation. It uses a transformer language model and is trained with musical data.

AudioGen MusicGen and Audio Gen generate audio from user text input, and the output is text-related. AudioGen is fed environmental voices and produces both environmental and musical sounds.

1.6.2 MuseNet

Open-AI built Muse-Net, which can generate 4-minute-long compositions with ten different instruments. Instead of using a wave technique, it estimates the next token in MIDI files to learn patterns of harmony, rhythm, and style. Its task works similarly to GPT-2, but instead of text as a sequence, it uses audio data for tokens in the sequences. Its transformer design has been trained to predict the next token in the sequence. The transformer's architecture is based on the magenta project's Wave2Midi2Wave concept.

1.6.3 Music Transformer

Music Transformer is a Google Brain model for creating lengthier pieces. Producing a long-duration piece of music or audio is a difficult challenge since a musical sequence contains numerous time scales. It has a similar architecture to RNNs, but instead of employing LS-TM, it employs Relative Attention, which is analogous to the Attention Mechanism in NLP tasks but more efficient for music production. This transfer-based methodology allowed us to easily access prior occurrences.

1.6.4 AudioLM

AudioLM is one of the language model which generates long term consistence sounds from text given and generated sounds have excellent audio quality. Speech generation tests show that AudioLM can not only produce syntactically and semantically coherent speech without any text, but the model sets are nearly indistinguishable from speech. Additionally, AudioLM can simulate arbitrary audio signals such as piano music and speech. The audio inputs are mapped to sequence of discrete tokens and in the representation space it acts like language modeling.

The two sorts of sound tokens are utilized by AudioLM. To begin with, from the self-supervised language demonstrated by w2v-BERT, semantic tokens are recovered. These markers capture both nearby connections and global long-term structure while impressively downsampling the audio source. Permit for expanded cluster demonstrating.

Be that as it may, the sound that was reproduced from these tokens is of poor quality. Acoustic tokens delivered by the SoundStream neural encoder are utilized to capture specifics of the sound waveform and empower high-quality union, in expansion to semantic tokens to induce over this confinement. High sound quality and long-term consistency are accomplished by preparing the framework to at the same time deliver semantic and acoustic tokens.

This model is not trained with any music text or symbolic representation. The information of the data is its waveform. There are 2 tasks that need to be considered, one of them is speech generation other one is piano generation.

1.6.5 MusicLM

MusicLM is the model which generates high-fidelity music from text given. The model can be conditioned by both text and sounds.MusicLM has ability to process music data to generate hierarchical sequence to sequence model, and it is able to produce several minutes long music at 24kHz.

1.6.6 Wave2Midi2Wave

Wave2Midi2Wave is one of magenta projects which uses transformers to generate music. It consider music data as language with symbolic representation of music such as notes and tabs. It coverts given wave sound data to midi format as its name Wave2Midi, and it extract symbolic representation. With the language model, it learns how should notes order, how to create harmony with symbolic representation then, with the given sound data, it starts to continue from the given one and tries to predict next note as like next word prediction task in NLP then it constructs midi data with generated notes and in Midi2Wave part, It is able to convert midi format to wave format with any kind of musical instrument.

2

Methods of Sound Generation Neural Networks

Audio creation becomes difficult for AI-based tasks because of its size. All of the research in this field indicates that there are numerous ways to make sounds by changing audio data into another type of data. The generation of audio using neural network architecture has become one of the models' issues. Many designs can distinguish and address certain problems, yet developing with only one model is frequently insufficient. Many models, such as GANs for successful image production, encoder-decoder techniques, and transformers for NLP applications, are presented to generate consistent and long audio with great quality.

The methods utilized for picture generation necessitate image similarity. As a result, audio data must be represented in image-like formats such as spectrograms. The model then operates as an image-generating task to build new image-like audio representations, and from there, the waveform must be predicted and audio samples are generated. To improve the quality of these models, some pre-processing and post-processing activities are required.

Language models as audio creation take a different technique; in this case, some tokens, similar to NLP tasks, must be generated. RNNs with LSTM models or transformer-based models dealing with audio data dimension. One approach of NLP models believes that music has its own language and that audio data must

be represented in text as in the NLP challenge. In the pre-processing stage, all audio data must be transformed into tab or note representation. Unfortunately, this approach has limits, even though they are quite successful in generating longer than 4-minute sounds, they can synthesize instrumental and musical audio data from the symbolic representation. In other words, they will not be able to generate bird noises, traffic sounds, or non-instrumental sounds.

Because of the limits of symbolic representation as a starting point, we must discover another way to employ it as a representation in the transformer. Based on models, it came up with the idea of using audio waveforms to make new audio.

2.1 Introduction to Neural Networks for Sound Generation

2.1.1 What is audio data ?

Audio data consists of 44.100 samples per second. Because images have 256 values, the audio number is 65.536. Audio data is very cyclical. While image data principle components include edge features, audio components analyze sinuses' cyclic ancient patterns. Millions of time steps are included in a typical audio file of a few minutes sampled at 44.1 kHz. Music representation is typically separated into two categories: symbol representations, which are discrete variables, and audio representations, which are continuous variables [18]. There are also different types of audio files that contain audio data information.

The acronym MIDI stands for "Musical Instrument Digital Interface," and it refers to a file format that represents musical data information as well as communication for electronic instruments, computers, and gadgets[18]. It does not include audio data such as MP3 and WAV. It includes instructions on how to play the music. It has notes to play, duration, and velocities (which specify how loud the note is played), as well as other types of control factors such as pitch bends, etc...

To convey precise instructions, data packets known as MIDI messages are employed. MIDI messages include notes and note messages, control change messages, and program change messages, to name a few. MIDI channels are used to segregate distinct MIDI data streams within a single MIDI connection. MIDI is a versatile and widely used format that is used for a variety of tasks, including music creation and production, live performances, and the transmission of electronic instruments. This reduces the need to deal with difficult audio waveforms, allowing musicians and producers to compose, modify, and share musical ideas.

WAV is a popular audio file format for storing digital audio files that is a subset of the Interchange File Format specification. It is widely used on Windows-based computers and in professional audio software. "Waveform Audio File Format" is abbreviated as WAV. Because of its high audio quality, WAV files are suitable for storing uncompressed or losslessly compressed audio data. The format does not compress the bit-stream and stores audio recordings with varying sampling rates and bit-rates.

Files with the.mp3 suffix are digitally encoded audio file formats based on MPEG-1 Audio Layer 3 or MPEG-2 Audio Layer 3. The Moving Picture Experts Group (MPEG) created it utilizing Layer 3 audio compression. The MP3 file format achieves compression that is one-tenth the size of .WAV or .AIF files. The format allows for the streaming of such audio files over the internet for online listening, which was previously not possible due to the enormous file sizes of the audio files. Parameter settings such as bit-rate, sampling rate, composite, or standard stereo can be used to modify the sound quality of an MP3 audio file.

2.1.2 I/O Representations

The fundamental challenge in developing AI-based music is translating the music into a format that machine learning models can interpret. [4]Numerous more sorts of melodic documentation exist in expansion to the staff documentation utilized in console music. For illustration, tablature (truncated "tab") may be a notation framework that speaks to instrument fingering instead of melodic notes. It's prevalent on strung rebellious like the guitar and ukulele, as well as free reed aerophones like the harmonica. The model must recognize informa-

tion as numeric vectors, music must be represented as a series of numeric tokens including rhythm, notes, timbre, and other important data elements. These tokens serve as musical representations that the AI system can process. Musical expressions are often classified into two types: symbol domain and audio domain. The nuances drawn from symbolic expressions are incredibly important in music and have a large influence on how people appreciate music.

This section will go over the audio data types that can be used with AI models. The most difficult difficulty with audio generation is input and output because models are quite successful at generating data that is comparable to input data. There is no single method for training input sets.

Symbolic Output

Piano rolls (image-like) One of the common inputs for the music generation models is Piano rolls. Piano-roll expresses the music score as a 2D matrix that can be regarded as an image. It is a matrix representation of music which includes information of notes and pitches as rows and time as columns. The value inside that cell represents the velocity of notes. The matrix is called a score-like matrix. The time information can be either absolute or symbolic. Absolute time is the note occurrence and symbolic time is the version of extracted tempo information.

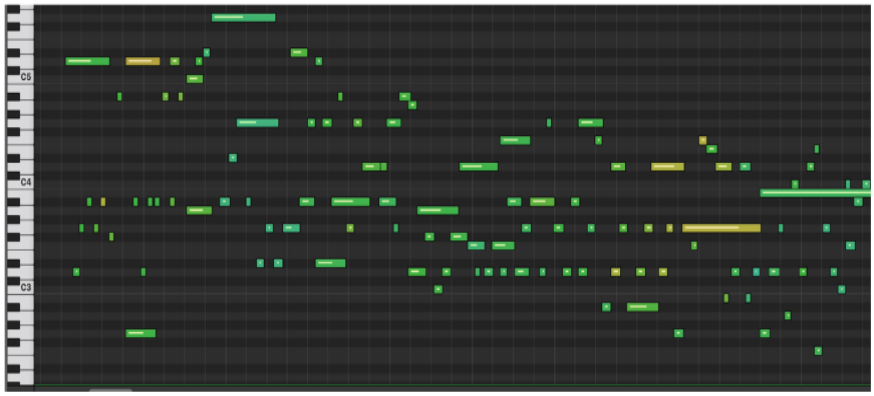


Figure 2.1: Pianoroll

There is also multi-track piano roll representation which includes many pianos to represent one specific track. If there are N tracks in the music piece then it will have N piano rolls. MuseGan and Lead Sheet arrangement projects using data sets that included piano roll data.

To increase performance, it converts MIDI data into DCGAN-friendly[18] piano roll-style graphic pictures and adds RGB channels as an additional information carrier. It can be portrayed in a similar way to a piano roll, however, the pitch dimension changes to reflect chord types and different components of the drum. The beat resolution further quantifies the temporal dimension.

Audio Output

Waveform A waveform is a graphical representation of a signal. Depending on the sort of input used to generate the wave, it can be sinusoidal or square. The waveform is determined by the parameters that define the wave's size and shape. In the [18] context of deep learning, sound output frequently refers to the final sound signal created by a generative demonstration. The waveform may become a widespread way to represent sound. A waveform displays the plentifulness of a sound source over time.

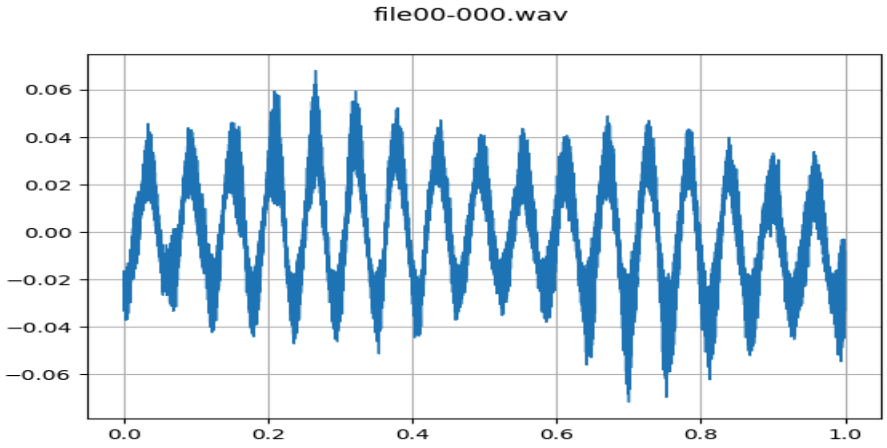


Figure 2.2: Waveform

The waveform [30] has two primary properties to digitize the signal in addition to amplitude. The sampling rate is the number of times a continuous signal is

sampled, while the bit resolution is the number of times a continuous signal is sampled. How many bits can be used to represent a signal in the -1 to 1 range.

Spectrogram Spectrogram is of representation of audio data[18]. The image-like illustration of present frequencies signals over time. They're useful for audio processing, speech recognition, music analysis, and even more scientific disciplines such as astronomy. According to typical manual features used for audio analysis, Spectrogram has more information than original audio.

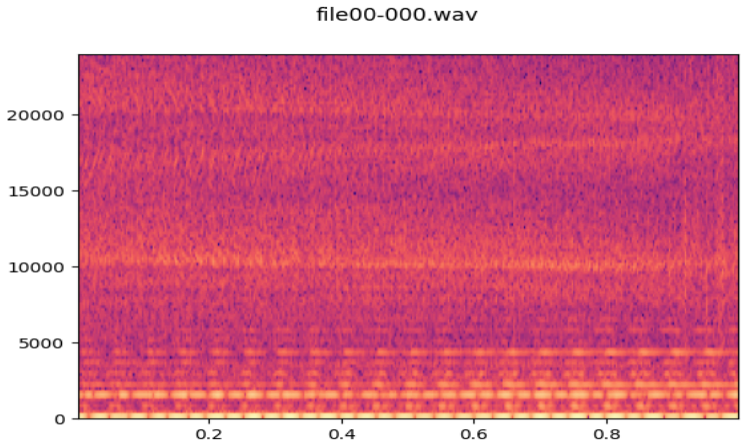


Figure 2.3: Spectrogram

Magnitude spectrogram It is an amplitude spectrogram [18] with frequency on the horizontal axis and frequency amplitude on the vertical axis. If it is to be utilized for audio generation, it must go through some reconstruction, which includes a variety of derivation approaches. Because it is not real-time, each time step takes the entire time scale of the signal. There are multiple iterations required for forward and inverse STFT.

Mel spectrogram Mel spectrograms differ significantly from conventional spectrograms that plot frequency as a function of time in two main ways. Mel is a transformation that transforms the actual measured frequency to a known center frequency. Convert a log-magnitude STFT spectrogram to a mel spectrogram. In an empirically consistent manner, the mel spectrogram reduces the

STFT spectral resolution. It follows that some data will be lost as a result. It is often used to train neural networks.

2.1.3 Generation

Symbolic melody generation

The target is creating the output of a sequence of discrete music events. MuseGAN and Midi-Net are examples of this generation. In MuseGAN inter-track and intra-track discriminative feedback have been proposed and multi-loss training has been developed to produce to train GAN to produce music tracks. The intra-track feedback is for generating individual tracks; meanwhile, the inter-track is for evaluating the joint performance of tracks like the band leader or composer. So different GAN models have been worked on, one uses inter-tracks for composing, one uses intra-track for jamming and the last is hybrid GAN which uses both.

However, the use of symbolic representations has limitations[9]: one of these is that the nuances abstracted away by representations are often musically quite important; for example, the precise timing, timbre, and volume of the notes played by a musician do not correspond exactly to those written in a score; and secondly, symbolic representations are often tailored to specific instruments, reducing their generality and implying that a lot of work is required.

Arrangement generation

The music arrangement is reconstructing and re-conceptualizing the process of creating the piece. Arrangement generation is one of the sub-tasks of automatic music generation. The generation process includes references from the original melody such as chord progress and other structural information. Usually data set includes MIDI format. The role of arrangement is being a bridge between the lead sheet, audio, and full score and connecting them. The arrangement must be style-consistent re-orchestration, must have precise time alignment of audio, lead sheet, or full score as a reference from original music and data-set must provide labels.

Style transfer generation

It is inspired by image style transfer. Music style has not been explained clearly in a scientific perspective, yet. They solve different problems in the computer music area. Style transfers are based on the idea of altering the style and keeping the content fixed. It separates and recombines music contents and styles. It ranges from compositional features to acoustic ones. Style transfer has different challenges, including different music areas according to interpretation. Those challenges occur by algorithmic composition, artistic performance, or sound synthesis.

The music content is abstracted information style is the information that increases interpretation and realization. the system must be flexible for dealing with different scales, able to gather the performance control and information of score from input and manipulate the representations of music.

There are three groups of style transfer, such as composition style transfer, performance style transfer, and timbre style transfer. Timbre style is directly related to sound generation and more importantly, works for generating different acoustic instruments sounds. It transfers and adjusts the timbre information and is applied to a sound representation. It saves the content information. Wave-Net auto-encoders and audio spectrograms neural style transfer systems are examples of this method. Meanwhile, Performance style transfer deals with adjusting control information and saving the information of implicit score content. It is related to performance rendering. It decodes control and notes and combines different controls to create a new musical character. we can consider composition style transfer for the musical genre which saves identifiable melody contour and adjusts other score features. It creates a new harmony. arrangement and variation.

2.1.4 Methods

Rule based methods

AI assumes or adopts established rules that are created by human-friendly language. Those rules give an idea of what to do later according to the task problems and are defined by human code such as adding an if-else structure. These

rules are affirmed belongings and practice. Rule-based frameworks consistently produce the same output, from given a variety of inputs making them unsurprising and reliable. Rule-based frameworks are essential for troubleshooting and tracking because rules are essentially written in a way that people can understand.

Rule-based manufactured insights frameworks utilize a set of inputs and a set of rules to deliver a yield. The framework to begin with chooses which arrangement to apply to the input. In case the run of the show is appropriate, the framework takes the fitting steps and produces the yield. In case no substantial enlightening is accessible, the framework may create standard yield, or incite the client for extra subtle elements.

In some cases, music generation with AI models is done by following some of the music rules and structures which are defined before. To create coherent compositions, music theory principles can be used.

Concatenation based methods

Concatenative sound synthesis (CSS) strategies use a large database of source audio divided into units and unit selection calculations called targets that determine how the units are placed. The perfect flavor for synthesized sounds and expressions. Selection is based on identifiers. unit. It is a reflection extracted from the source audio or a high-level identifier written into it. You can flip and combine selected units to perfectly align your targets. However, if the database is large enough, it is more likely that a coordinator can be found, reducing the need to make changes even if the sound quality is constantly changing. Units may not be uniform. This can be anything from audio clips to instruments, notes to complete phrases. However, uniform estimation and classification units are often used, and sometimes units similar to flag intervals, along with ghost checking or a combination of overlap and addition.

Regular sound synthesis strategies are based on displaying sound flags. Building a show that properly brings out all the subtleties of sound can be a painstaking process. Concatenated synthesis, on the other hand, completely blocks out these subtle elements by using the original recording. For example, unit settings from the database are taken into account when selecting units, allowing you to

synthesize highly realistic-sounding trains. In this data-driven approach, rules start from the information itself, rather than being created with careful consideration as in a rule-based approach. Concatenation mixes can be more or less data-driven. More is invaluable because the data contained in the many sound illustrations in the database can be misused.

Machine learning based methods

The use of machine learning in audio and music signal processing is growing rapidly. Machine learning methods already dominate emerging approaches in the disciplines of Music Information Retrieval and Audio Signal Processing. While monophonic audio signals have been largely handled, significant issues in polyphonic pitch transcription remain. Machine learning-based methods, such as statistical parametric methods, are less rigid and allow for things like merging data from numerous speakers, model modification using minimal quantities of training data, joint modeling of timbre and expressiveness, and so on. Probabilistic modeling is one of the most used methodologies.

There have been many researches on chord generation based on traditional machine learning methods. While monophonic audio signals have been largely handled, significant issues in polyphonic pitch transcription remain. The commonly used models are the Markov model, and the hidden Markov model (HMM)[18].

2.2 Music Representations and Neural Networks

The music representations which are discussed can be used for different kinds of methods. So far with the neural network perspective, with the feature of sound-like being highly dimensional and complex structure, there are many approaches to processing sound data. As we know the GAN models are quite successful methods for image generation, and for generating music, they are getting benefits of musical features that can be represented in 2D space. For instance, converting 1D wave data to 2D spectrogram data which is signal over time at various frequencies, and trying to generate a similar 2D matrix then predicting waveform from the generated spectrum.

There is another way that we can consider music data as language either with symbolic representation of it or waveform and processing musical features as like text in tasks NLP. Because from notes to velocity, it can be considered as the way of speaking in musical language. So as in the speech generation task, the information is gathered from musical features.

Even while language models are quite capable of generating extended-length coherence audio using transformers or LSTMs, there are several obstacles and limitations. These methods work great for instrumental music, but when we wish to make noises like dog barking or traffic noise, it is not possible to capture these types of sounds in text-like notation. So, in order to generate this type of data, the waveform of the data must be processed.

2.3 Data-set

2.3.1 Audio Craft

The data set takes a unique approach because the audio craft processes not only audio but also language. This is why each piece of musical data must include a description, some keywords, genre, instrument name, moods, and metadata.

Each audio recording requires its own json file with information such as a description, keywords, and name. The JSON files are compressed as data.jsons or data.jsons.gz files and include one JSON per line. This JSON contains the audio file's location as well as any associated metadata. They are included as data source subconfigurations in the configuration. It contains links to the manifest file URL and essential metadata for each AudioCraft phase.

The AudioCraft dataset is not publicly available, although the model can handle both mp3 and WAV files.

2.3.2 Food-Sound

The data collection contains 5500 sounds with JSON file descriptions that may be used to generate tokens for usage in MusicGen and AudioGen designs.

These descriptions are used for sound matching. There are 22 descriptions that correspond to the 5500 data points. The metadata json, which comprises the directory, filename, and sample rate, is created from the audio data.

The following are the descriptions used for the food sound project:

- Continuous glitchy elements with moving panorama
- Continuous hum with pitch and frequency modulation
- Continuous, deep and spatial, stable and rumbly
- Continuous, deep, spatial and gloomy
- Continuous, hollow, humming with ascending and descending pitch
- Continuous, humming with regular low impulse
- Continuous, rapid and glitchy elements with long pitch envelope
- Continuous, rapid elements with slowly moving panorama
- Continuous, rapid, glitchy buzzing elements
- Continuous, rapid, sizzling elements
- Continuous, scary sizzling cymbals with varying pitch
- Continuous, soft noise. Alarm like effect
- Continuously stridulating, buzzing and sizzling sound of processed crickets
- Digital, continuous humming with varying pitch
- Light, a tonal ring of shredded glassy elements like insect swarm
- Sci-fi, science fiction, continuous, slightly sizzling elements
- Sci-fi, science fiction, squishy humming,
- Spatial, continuous, glitchy element with modulation and panning movements
- Spatial, deep, synthetic with continuous frequency modulation
- Subtle, softly sizzling element with moving panorama
- Synthetic sci-fi, science fiction with a sixth interval and slight moving panorama
- Synthetic sci-fi, science fiction with glitchy, slowly varying pitch envelope and heavy low rumbling

2.4 Deep Neural Network

2.4.1 WaveNet

WaveNet[12] is a deep generative technique that synthesizes high-quality audio waves. It generates sound from the raw audio waveform and results successfully in excellent outputs. It manages the conditional probability distribution of the next sample from the previous one . It was inspired by neural autoregressive generative models. In autoregressive architecture, each waveform sample is produced based on previous samples. In this way, it can capture sophisticated dependencies, and that allows it to reproduce remarkably realistic audio. Initially, it has been developed for text-to-speech and general audio generation.

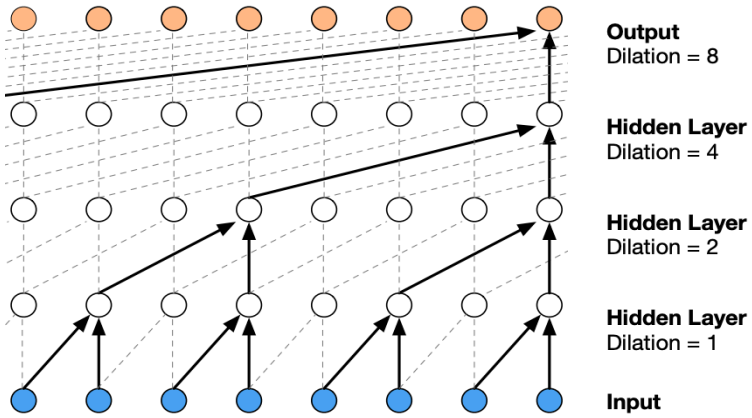


Figure 2.4: WaveNet Architecture [22]

The main components of the article are causal dilated convolutions, which ensure that the model only looks into the previous samples to generate the future sample at each time step[30]. A stack of convolutional layers is used to model the conditional probability distribution[22].WaveNet relies heavily on causal convolutions. The model cannot break the ordering of the data by utilizing causal convolutions. This is easier to accomplish with 1-D data, such as audio, by moving the result of a conventional convolution by a few time steps. Dilated convolution is also known as holes convolution or a-trous convolution. The kernel varies linearly in ordinary convolution (dilation = 1). It is equivalent to

a convolution with a bigger filter, in which the initial convolution is filled with zeros to enhance the network's receptive field.

2.4.2 RNNs

A recurrent neural network (RNN) is a deep learning network design that predicts time series or sequential data. For the usage of RNNs in music generation, there are many approaches such as Hierarchical Recurrent Neural Network (HRNN), VirtuosoNet, and CVRNN. Style performance model based on RNN and Siamese network focusing only on note dynamics autoregressive model learned directly during performance MIDI notes contain timing and dynamics expressive force [17]. Those models deal with symbolic representations because of their usage in language models as roots. In theory, an RNN can learn the temporal structure of any length in the input sequence, but in practice, learning long-term structure becomes increasingly difficult as the sequence lengthens. The length of the learnable structure in each model is limited, and this limit is determined by the complexity of the sequence to be learned.

HRNN [31] is made up of three LSTM-based sequence generators: the Bar Layer, the Beat Layer, and the Note Layer. For learning melody, the Bar Layer and Beat Layer are trained to yield bar profiles and beat profiles, which are supposed to capture the high-level temporal aspects of melody.

The Bar Layer and Beat Layer produce a melody piece with a length of one bar, and the Note Layer is trained to generate a melody conditioned on the bar profile sequence and beat profile sequence. With the previous bar profile, $B_{t_{16}}$ as input, the Bar Layer creates a bar profile B_t first. The Beat Layer then creates 4 beat profiles $b_t, ..., b_{t+12}$ based on the bar profile B_t) using the input b_{t_4}. The Note Layer depends on both B_t and b_t in order to generate the notes $n_t, n_{t+1}, ...n_{+3}; n_{t+4}, ..., n_{t+7}$; and so on.

The Performance RNN, on the other hand, is a variety of RNN for sound creation that creates expressive timing and dynamics using a stream of MIDI events. A fundamental component of MIDI is a series of precisely timed note-on and note-off events, where each one determines the pitch of the subsequent note.

Additionally, velocitythe force with which the note is struckis a note-on event. [25]

The fundamental issue with RNNs is that they can easily lose their knowledge of the past if the weights in the feedback loops are not correctly calibrated, making created music uninteresting. Due to the gradient vanishing issue, it is challenging for RNNs to preserve extensive historical information about sequences. Long Short-Term Memory (LSTM) was introduced two decades ago for language pricing and sound generation. It is one of the strategies that protect RNNs from memory loss by gating off the results accumulated by the RNNs and storing them in a memory cache.RNN architecture-LSTM is used to help the network remember and recover information in the sequence in order to address this issue.

2.4.3 VQ-VAE

Music style transfer and symbolic representation of music can both be accomplished using VQ-VAE approaches. VQ-VAE with multi-scale discrete features can produce high-quality images comparable to most of GAN models. There is research on using VQ-VAE for music generation [14].In variational autoencoders, VQ-VAE introduces a hierarchical framework. It combines the benefits of VAEs and autoencoders to produce a more expressive and economical model.

Vector quantization (VQ) is the process of transforming continuous or discrete data into vectors, with the purpose of achieving data compression by quantization.

A neural network an auto-encoder(AE) has one hidden layer with the same number of output nodes as input nodes. They have a network of encoders to parameterize the posterior distribution $q(z|x)$ of discrete latent random variables z given input data x, and a decoder with a distribution $p(x|z)$ over the input data [21]. These are together with VQ-VAE with discrete latent variables with a new way of training, inspired by vector quantization to learn useful discrete representations in a completely unsupervised way. [18]

The embedding layer with two convolutional layers of kernel size is responsible for reducing pitches to dense representations, while the quantizing layer is responsible for transferring continuous latent features to discrete ones in VQ-VAE. By forward-passing the training set to the dictionary, obtained quantized indices k from the VQ-VAE encoder and trained fresh 2-layers LSTM with them. The encoder maps the input x to a sequence z of discrete codes from a codebook, and the decoder tries to map z back to x [5]. Because we are employing d-dimensional space and a codebook of k with the rest of the model parameters, the complexity of VQ grows quickly, necessitating the use of a more practical quantization technique.

2.5 Types of Generative Adversarial Neural Networks for Sound Generation

GANs have two networks: a generator and a discriminator. The generator generates new samples from latent code, but the distribution of signals and new samples cannot be distinguished from the training distribution. The generator is the primary goal; the discriminator can be abstracted afterward because it is only used to improve the generator. Adversarial is a game played between two people. It is one of the unsupervised techniques for mapping low-dimensional latent vectors to high-dimensional data.

2.5.1 DCGAN

Deep Convolutional GANs[23] are the extinction of GANs architecture by using convolution and convolution-transpose layers in the discriminator and generator. Discriminator includes a convolution layer, batch normalization layer, and Leaky Re-LU activation. The input is a matrix, and the output is the scalar probability of the inputs real distress provided the generator has batch norm layers and Re-LU activation. The input is a latent vector that is created from a standard normal distribution and the output is a dimensional matrix.

The main architecture of DCGAN for stability is that any pooling layers are replaced with stridden convolutions for the discriminator and fractional-strides convolutions for the generator.

LSUN scene modeling is applied for DCGAN which is for avoiding overfitting and memorization of training data. Instead of data augmentation with this method model is trained with LSUN bedrooms data set to illustrate how to model scales with more and better resolution generation.

WaveGAN [10] GANs are well-known for their powerful approach to image generation. The spectrum can be one of the viewpoints to interpret an audio signal as image data for audio production, however, there is a difficulty with the non-inevitability of the perceptually-informed spectrum. The GAN method allows for the quick and easy sampling of enormous volumes of audio.WaveGAN is capable of synthesizing a second slice of raw waveform audio. WaveGAN is comparable to DCGAN-based GAN for Images. Without labels, it can learn to make tiny sentences and synthesize audio from domains such as percussion and piano. It is one-dimensionally processed and aggregates result in a two-dimensional analog by flatting DCGAN.

It replaces pooling layers with striped convolution and fractional striped convolution. It also uses batch normalization and remove fully connected hidden layer for deeper architecture as has been mentioned in DCGAN. Besides that WaveGAN uses 2D convolutions to flatting, for instance 5x5 2D convolution becomes 25 1D convolutions and it increases the stride factor.

Phase Shuffle Phase shuffle is the technique that regularizes the discriminator so that it doesnt just focus on really low-level details in the generator like having a certain wave be of by four frames and using that to discriminate the generated and real audio samples.

SpecGAN The Spectrogram Generative Adversarial Network (SpecGAN) is a deep learning model that focuses on spectrograms to create audio signals from spectrogram representations of audio input.

The raw audio data is first transformed to spectrogram pictures, which are representations of frequency across time. To develop SpecGAN[10], audio is processed using the short-time Fourier transform with 16ms windows and an 8ms stride. This technique produced 128 frequency bins separated linearly between 0 and 8 kHz. The magnitude of the generated spectra is determined,

and the amplitude values are logarithmic-ally scaled to better align with human vision.

The magnitude of the resulting spectra is taken, and the amplitude values are scaled logarithmic-ally to better accord with human vision. Each frequency bin is normalized to have a zero mean and unit variance. The spectra are trimmed to three standard deviations and re-scaled to [-1,1]. This stage generates the input necessary to train a GAN. To generate waveforms from the resulting spectrograms, invert the spectrogram preprocessing stages, resulting in linear-amplitude magnitude spectra.

2.5.2 Conditional GAN

Conditional Generative Adversarial Networks (cGAN)[20] are one type of GAN in which the model generates images or audio conditionally. As the tradition of GAN, the generator learns how to produce a new image or audio and the discriminator tries to distinguish between the generated one and the real one. For both the generator and discriminator, there is a condition which is a kind of extra information applied to them. The model learns multi-mapping with the inputs and outputs which have different contextual information.

By having extra information, it has faster converges and it is easy to control the generator's output with the labels given.cGAN is not closely unsupervised learning because it needs some label information as input in an additional layer.

2.5.3 Cycle-consistent GAN (CycleGAN)

CycleGAN[3] is a successful neural domain transfer architecture for graphics that can move polyphonic musical compositions from a source genre to a target genre, such as from jazz to classical, simply by changing the pitch of a note. It has additional discriminators to balance the power of domain transfer versus preserving the content of the original input, as well as separate gender classifiers to measure the influence of genre change. The audio samples provided illustrate that genre switching can not only be detected by a neural network classifier but can also be heard by humans.

Domain transfers are attractive because they require the development of unique representation-learning methods that can be applied to other areas of deep learning research. Domain transfer requires neural network models to have a deep understanding of the underlying domain. Neural style transfer utilizing deep generative models is an important component of deep representational learning research.

The terms style and domain transfer are sometimes used interchangeably. In the context of neural networks, style transfer often refers to applying the explicit style attributes of one image while preserving the quality of the explicit content of another.

This model, on the other hand, is built on a generative adversarial network (GAN) and includes a vanilla GAN as generator G and discriminator D. A generator tries to generate real-looking data from noise, whereas a discriminator tries to understand the generator's output from real data. In a two-player minimax game, G and D are learning iteratively. Because the model's objective is to transport music from one domain to another, the generator receives real samples from the source domain as input rather than real noise.

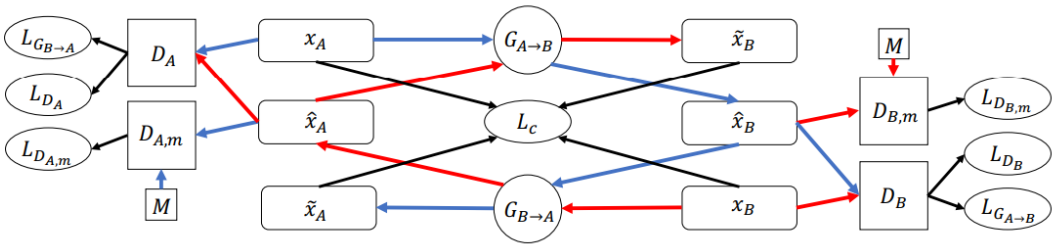


Figure 2.5: Architecture of CycleGan [3]

The architecture of the model can be seen in Figure 2.5. The model's procedure is to exchange samples from A to B and bad versa; it has the same structure as the late discharged CycleGAN, which comprises two GANs stacked in a cyclic way and prepared in harmony. One generator transports information from space A to space B, whereas the other exchanges information from space B to

space A. Each generator yield has one discriminator tied to it. Blue and red arrows demonstrate space exchanges in restricting bearings, though dark bolts demonstrate misfortune capacities. L2 norm for the adversarial loss for the generators;

$L_{G_{A\to B}} = \|D_B(\hat{x}_B) - 1\|_2$[3]

$L_{G_{B\to A}} = \|D_A(\hat{x}_A) - 1\|_2$

L1 norm for the adversarial loss which is cycle consistency loss

$L_c = \|\hat{x}_A - X_A\|_1 + \|\hat{x}_B - X_B\|_2$[3]

The structure of the cycle consistency loss ensures that input is matched to itself back over the cycles between the generator and discriminator. It is utilized to prevent the loss of information from the input to the generator; it aids in the preservation of that information for back-transformation.

Since the model needs to differentiate between song genres in order to develop songs based on genre, rather than just playing sounds, the loss needs to be calculated for that classification. This feature may cause the training phase to be unstable with this CycleGan sound generation. However, stability must be balanced during the training phase. The generator also needs to learn genre translation from source to target. In this situation, the discriminator must also understand genre transitions. This model includes two different identifiers to avoid genre patterns that lead to deception. As a result, these two identifiers can make sense of the differential data from the target domain. Discriminator loss will be in the end,

$L_G = L_{G_{A\to B}} + L_{G_{B\to A}} + \lambda L_c$[3]

λ is used as a weight for the cycle consistency loss. For the discriminator loss will be,

$L_{D_A} = \frac{1}{2}(\|D_A(x_A) - 1\|_2 + \|D_A(\hat{x}_A)\|_2)$[3]

$L_{D_B} = \frac{1}{2}(\|D_B(x_B) - 1\|_2 + \|D_B(\hat{x}_B)\|_2)$[3]

$L_{D,all} = L_D + \gamma(L_{D_{A,m}} + L_{D_{B,m}})$[3]

γ is represents weight for extra discriminators.

Due to dimension issues, this methodology shows that methods can be combined depending on the sound generation task. Unfortunately, one method is not always suitable for processing specific audio data. To obtain more accurate and consistent results, a different approach to handling these characteristics of audio data is required.

Progressively growing of GANs (PGGAN)

GANSynth [11] A method for producing high-quality audio using Generative Adversarial Networks and the NSynth database. Most audio waveforms, unlike visuals, are extremely periodic, such as speech and music. Convolutional filters trained on this data for various purposes frequently learn to construct logarithmically scaled frequency selective filter banks spanning the human hearing range.

It uses progressive growth of GAN architecture with 2D convolution, which aims to grow the generator and discriminator progressively. The outputs of the model are mel-spectrogram and instantaneous freq (IF). It uses IF to drive the phase and then inverse STFT to gather the waveform. The inputs of the generator are 256-dim random vector Z and 61-dim one-hot vector (MIDI 24-84) for pitch conditioning. Auxiliary pitch classification is loss for the discriminator to the real/fake loss and tries to predict the pitch label WaveNet solves scale problems by centering the finest scale possible and hardly external conditional signals for global form. But it causes slow speed for producing samples.

2.6 Language Models

2.6.1 Transformer-Based Audio Generation

Transformers attempt to tackle the parallelization challenge by combining encoders and decoders with attention models. The model's ability to translate

from one sequence to another is accelerated by attention. Transformers are multi-encoding, multi-head architectures with attention and multi-head mechanisms. They are without a doubt a powerful and efficient methodology for Natural Language Processing (NLP) problems. Additionally, they have recently been introduced to be used for image generation, image classification, audio generation, and audio classification jobs. They attempt to tackle the parallelization challenge by combining encoders and decoders with attention models. The model's ability to translate from one sequence to another is accelerated by attention.

The basic idea behind transformers is to create self-attention mechanisms by using positional embedding to avoid consuming memory as LSTM. As a result, they allocated fewer resources than RNNs with LSTM, making them faster and more efficient.

The Transformer [] is based on through and through position representations, which are included in pre-position input representations by means of positional sinusoidal or learned position embedding. The relationship between musical notation (score) and actual sound is similar to the relationship between text and audio. Timing and pitch are two illustrations of melodic measurements where relative contrasts show up to be more imperative than absolute values. To capture such pairwise relations between representations, display a relation-aware shape of self-attention that they successfully utilize to adjust self-attention by evacuating the introductory Transformer between two positions, tests from a Transformer with our relative thought component, and the normal timing arrangement appeared inside the dataset.

They were able to get compelling results in music era interchange by learning transformers on inactive representations and conditioning a WaveNet generator, which would have been impossible without the direction of meta-data and convolutional designs [29]. They have also been used to learn idle sounds. Transformers use a consideration instrument with the yield at one location dependent on the input at another. There are several different transformer types. They all share the same basic attention layers, but some are more specialized for specific tasks.

Seq2Seq A variety of related attempts have been made to use neural networks to handle the general problem of sequence-to-sequence learning. Training models that translate sequences from one domain (such as English sentences) to sequences from another domain are used in sequence-to-sequence learning. The model[27] takes the input sentence 'ABC' and creates the output sentence 'WXYZ'.

The Seq2Seq task takes a sequence of audio frames as input and outputs a sequence of symbolic tokens representing the notes being played. [13].It can be used in sound generation tasks because it is employed in the LM model. An acoustic modelling[33] approach for speech conversion based on a framework of sequence-to-sequence neural networks. An encoder network translates input characteristics into a hidden representation, an attention module selects the encoder output, and a decoder generates acoustic features for each frame are all part of the model. There is also a post-filter network that enhances the altered acoustic features. Finally, the time-domain waveform is recovered using a speaker-dependent WaveNet. This paradigm eliminates the requirement for source filtering assumptions by using mel-scale spectrograms as auditory features for speech generation.

Because Seq2Seq excels at language translation, you may use it to convert melodies to chords and vice versa if your melody includes harmony or a chord progression. The code is received by the encoder side and passed from one side to the other. Similarly to a language task, each output of the final encoder is an input for each level of the decoder, with the coding shifted from the preceding output. To put it another way, chords are converted into melodies. The encoder includes a bidirectional attention mechanism, whereas the decoder has a single-stack forward addressing layer.

A sequence of inputs is mapped to a sequence of learned embedding plus fixed positional embedding in T5 architecture [13]. In terms of sequence length for the self-attention blocks, one constraint of sequence models applied to audio is that most audio sequences are too long to fit in memory when modeling using a Transformer architecture.

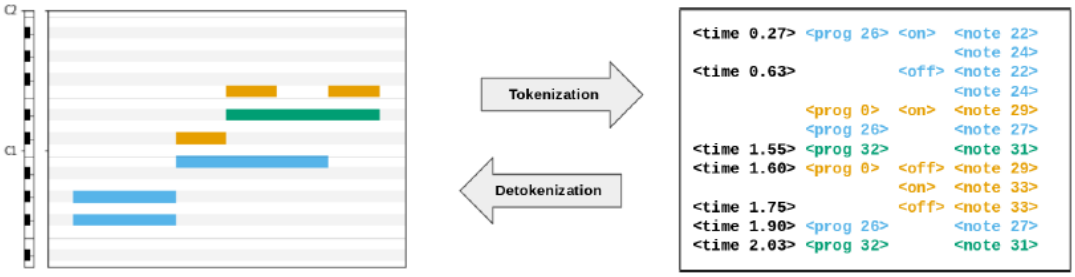


Figure 2.6: Tokenization/detokenization [13]

Transformer-XL Transformer-XL[7], which extends learning reliance beyond a fixed length while maintaining temporal coherence. As specific as its task in music generation we can say there are a few notes to make into a song, and Transformer XL excels at sequence development and auto-completion of song ideas. It improves on Transformer by introducing the concept of repetition and changing the position encoding mechanism. The iteration technique enables the model to use information from previous tokens in addition to the current training segment, thus extending the history[16].

Transformer-XL is a forward-directional decoder with memory and relative positional encoding for exceptionally quick and precise predictions. Relative sinusoidal positional embeddings are used in Transformer-XL. There is no restriction on the length of the sequence when padding to the left or the right. Similar to a standard GPT model but with the addition of a recurrence mechanism for two successive segments. A segment is a group of tokens that are sequential and may span several documents. Segments are supplied into the model to train it to pay attention to information in both the previous and current segments.

It is related to positional encoding and hidden state memory is included. Transformer Memory allows for extremely quick inference. Instead of re-evaluating the entire sequence on each prediction, you only need to assess the most recently predicted token. Previous tokens have already been saved in memory.

The key distinction between Transformer[16] and Transformer-XL is the use of features from segments earlier to the current segment while updating the

model parameters based on that current segment. The segments are from the same input MIDI sequence of a music piece.

Relative Positional Encodings Edges can capture information about the relative position differences between input elements in linear sequences [24]. Maximum relative position is thought to be clipped to a maximum absolute value of k, implying that beyond a certain distance, exact relative position information is useless. By limiting the greatest distance, the model can generalize to sequence lengths that were not experienced during training.

Multi-Head Attention [28] A query and a set of key-value pairs are mapped to output by an attention function, where the query, keys, values, and output are all vectors. The output is generated as a weighted sum of the values, with the weight allocated to each value determined by the query's compatibility function with the relevant key.

BERT stands for Bidirectional Encoder Representations from Transformers which is a bi-directional encoder that produces SOTA results in answering questions and filling in the gaps, token masking, and exceptional context. Series of Transformer encoder blocks, each with a multi-head bidirectional self-attention layer followed by a feed-forward layer. Each encoder block has residual connections and layer normalization[8].

It is used for music or sound generation when there is a song has been completed, but something does not sound quite right. A more constant rhythm is required for this song.BERT is an expert at filling in the blanks. To produce a new variation, remove sections of the song and apply BERT.

Audio Tokenization The same sequence continuation principle that is important to language models can be applied to audio data. The concept of audio continuation proposes that an input audio waveform can be regarded as a succession of "audio tokens." A model might then be taught to generate an audio continuation that matches the input's properties.

An audio sequence typically contains semantic characteristics, such as melodic/harmonic consistency in music, and Acoustic aspects, such as the distinct

tone of a voice or the timbre of a musical instrument. These dependencies, which can be semantic or auditory in nature, are critical for recording recurrent or developing patterns throughout time.

Tokens are intended to represent long-term structure in audio with a BERT-type language model trained on audio data that creates semantic tokens that capture both local interdependence and global long-term structure for the Audio Embedding w2v-BERT. Based on SoundStream, Audio Quantization provides acoustic tokens that capture the nuances of the audio waveform and enable high-quality synthesis.

2.6.2 Generative models of raw audio signals

The limitation of symbolic representation has been mentioned, so to go around these limitations, music can be displayed within the crude sound space instep [9]. Since computerized representations of sound waveforms are still lossy to a few degrees, all musically important data is held. Sound waveform models are moreover distant more wider and can be connected to recordings of any set of rebellious, as well as non-musical sound signals like discourse.

It has been modeled[30] for the probability distribution of the samples of the waveform of an audio signal when the decision is made to digitize the signal, the sampling rate, which indicates how frequently the continuous signal is sampled, and the bit-resolution, which measures how many bits are allocated to represent the signal in the range of -1 to 1. The WaveNet implementation to model technique with traditional neural network-based autoregressive WaveNet architecture is the same as the original, but in this study[30], the dilation rate is increased by a factor of 2 in each layer, totaling 10 layers, with 128 convolutional filters in each layer. The model is made up of three layers of causal dilated convolutions with a filter size of two that are dilated by a factor of two in each succeeding layer.

Also, in order to implement the transformer on the input data, heavy data augmentation such as amplitude scaling, time shifts, and so on was performed. Before delivering it to the Transformer architecture, the input one hot representations matching the level of the bit representation were sent via a positional

encoding layer to add positional information. The purpose of both the WaveNet and Transformer architectures is to forecast the next sample.

2.6.3 Language Modeling Approach

Audio signals span different scales of consideration, including speech, music, and natural sounds. For example, language can be analyzed at a very close acoustic or phonetic level, but also in terms of prosody, language structure, language use, or semantics. Music also follows a long-term structure but is made up of highly unstable acoustic signals. When it comes to sound fusion, these different scales are so interconnected that achieving high sound quality while maintaining high consistency remains a challenge, especially in the absence of thorough monitoring.

Using strategies such as propagation, adversarial training, and autoregressive waveform modeling, speech synthesis models delivered signals with a level of confidence that approximates reality. However, advanced models like WaveNet actually produce babble-like unstructured sounds without significant conditioning. On the other hand, language models have been shown to be able to represent long-term, high-level structures of data in a variety of formats, and the resulting advances in content and image rendering have allowed us to create reliable and reasonable common sounds.

A single channel audio sequence xR^T is processed. In the framework of AudioLm, the first tokenizer model maps x into a sequence of discrete tokens $h = enc(x), h = (h1, ..., h_T)$ from a finite vocabulary $T' \ll T$. Then Transformer language model which includes only a decoder processes these discrete tokens to y [2].

During derivation, the demonstrator autoregressively predicts the token sequence $\hat{h}$. The detokenizer model maps predicted tokens to audio for generating waveform $\hat{x} = dec(\hat{h})$.

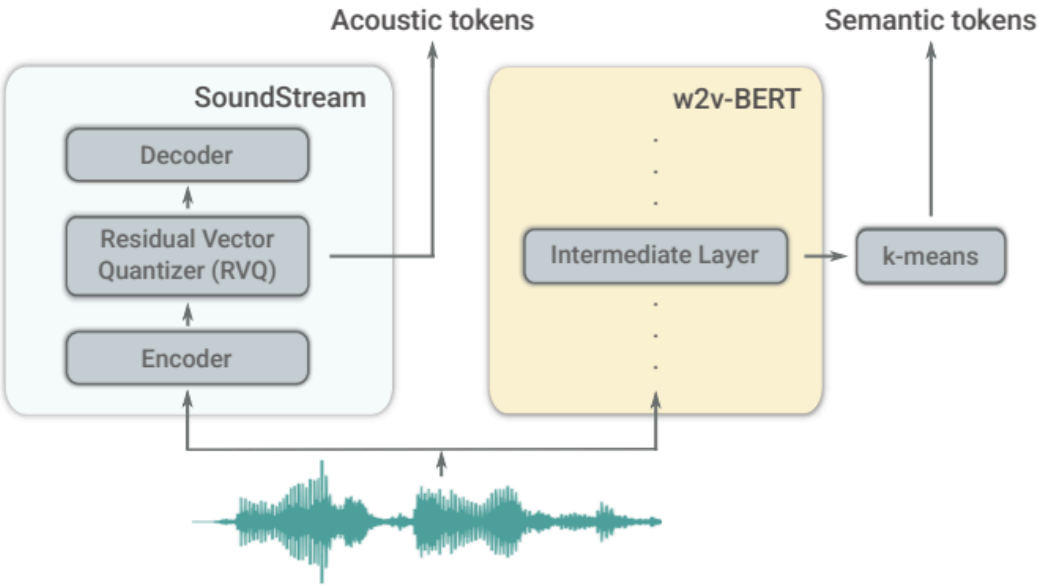


Figure 2.7: Overview tokenizers [2]

One of the advantages of having two models tokenizer and detokenizer is generating discrete audio representation. Not only generating high-quality audio is aimed but also long-term coherency is important to regenerate waveform. Both have semantic and acoustic tokens to combine, promising for the model to generate. This structure can be seen in figure Figure 2.7. While SoundStream generates acoustic tokens, the intermediate layer of w2v-BERT generates semantic tokens and ensures long-term structural consistency.

An input waveform is converted by SoundStream's convolutional encoder to a series of embedding at a sample rate that is considerably lower than the sample rate of the original audio. For a 16kHz input waveform, SoundStream is configured to generate 50Hz embedding. Each embedding is discretized using a residual vector quantizer (RVQ). RVQ consists of a hierarchy of Q vector quantizers, with each quantizer having a vocabulary of N symbols. SoundStream's convolutional decoder reconstructs the waveform by mapping this discrete representation to real-valued embeddings.

w2v-BERT, a model for learning self-supervised audio representations, is used to compute semantic tokens. AudioLM uses the representations of the pre-trained w2v-BERT to model long-term temporal structure in a generative framework, despite the fact that this model may be adjusted for discriminative

tasks like speech recognition or speech-to-text translation. In order to achieve this, an intermediate layer of the w2v-BERT MLM module is chosen, and embeddings are computed at this level. k-means on these embeddings with K clusters and utilize the centroid indices as semantic markers. Prior to clustering, normalizing w2v-BERT embeddings so that each dimension has a zero mean and unit variance greatly enhances their phonetic discriminability. w2v-BERT performs downsampling along the temporal dimension.

Different features of acoustic tokens acquired from SoundStream and semantic tokens obtained from w2v-BERT are obtained by comparing them in terms of audio quality to motivate the hybrid tokenization technique. By teaching a SoundStream decoder to rebuild audio from tokens, you may assess the accuracy of the reconstruction. Only a decoder Transformer on the series of acoustic tokens by flattening Y in a row-major order to a series of tokens y+o of length $T_A \cdot Q$, where $o = (o_1, o2, ., o_{T_A \cdot Q})$ is the vector of offsets for generating distinct token indices for the Q layers of the residual vector quantizer, with $o_i = (i_1 mod Q) \cdot N$.The model, which has only been trained on acoustic tokens, samples speech continuations every four seconds.

The semantic tokens for the entire sequence are first modeled using a hierarchical technique, and these tokens are then used as conditioning to forecast the acoustic tokens. This approach has two main advantages: first, the token grouping per organize is reduced compared to options like modeling the interleaves arrangement of semantic and acoustic tokens, allowing for computationally more efficient analysis. Second, the hierarchical modeling reflects the conditional independence suspicion that semantic tokens are anticipated to be conditionally independent of past acoustic tokens given past semantic tokens, that is, $p(z_t|z_{<t}, y_{<t})p(z_t|z_{<t})$

To capture long-term temporal structure in semantic modeling, the first stage uses $p(z_t|z_{<t})$, the autoregressive prediction of semantic tokens. Coarse acoustic, modeling Similar to the first stage, the second stage predicts solely the acoustic tokens from the coarse Q SoundStream quantizers based on the semantic tokens. The acoustic tokens in SoundStream have a hierarchical structure because of residual quantization. To handle their hierarchical structure, on the straightforward method of flattening the acoustic tokens in a row-major

order. Since y_1^1 is the first token predicted during training, the second stage models $p(y_t^q|z, y_{<t}^{\leq}Q, y_{<t}^{<}q)$ for $q \leq Q$ where the matching token sequence is $(z_1, z_2, ..., z_{T_S}, y_1^1, y_1^2, ..., y_1^Q, y_2^1, y_2^2, ..., y_2^Q, ..., y_{T_A}^Q)$.

Fine Acoustic Modeling The third configuration uses coarse Q′ tokens as conditioning to model the conditional probability distribution $p(y_t^q|y^{\leq}Q', y^{\geq}Q', y_t^{<}q)$ when q > Q. This operates on acoustic tokens that are compared with a fine quantizer. In other words, all the tokens associated with the coarse Q′ quantizer, the fine Q′ quantizer of the previous time step, and the token currently being decoded at the current time step compared to the coarse quantizer are used to predict $y_t^{<}q$. Sound quality is significantly improved by eliminating the outdated elements of lossy compression that remain after insertion.

After training, use AudioLM to generate audio, depending on the conditioning signal used, different forms of generation are achieved. All semantic tokens $\hat{z}$ are sampled in an unconditional generation scenario and used as conditioning for acoustic modeling. To create an acoustic token in an acoustic production environment, a semantic ground truth token z is obtained from a test sequence x as conditioning. The resulting audio sequences still have different speaker identities, but the content of the spoken sentences is always the same and consistent with the ground truth transcript of x. This shows how semantic tokens can effectively represent semantic content. The main use of Interest is to generate a continuation from a short prompt x.

To accomplish this, the prompt is first transformed into appropriate semantic tokens $z_{>t_s}$and coarse acoustic tokens $y_{\leq Q'}^{\leq}, t_a$. The first stage creates $z_{>t_s}$ts. It is a continuation of semantic tokens that are autoregressively generated based on the conditioning $z_{>t_s}$. In the second stage, the complete semantic token sequence $z_{\leq t_s}, z_{>t_s}$ is concatenated with the coarse acoustic tokens of the prompt$y_{\leq Q}^{\leq}, t_a$ and fed to the coarse acoustic model as conditioning. The coarse-grained acoustic model then samples the relevant acoustic token continuations. The fine acoustic model is used in the third stage to process the coarse acoustic tokens. Then use the SoundStream decoder to reconstruct waveform x using the prompt and sampled acoustic tokens.

2.6.4 Generating Music From Text

Although the ability to generate audio from such crude subtitles represents a significant advance, these models are still only capable of simulating basic acoustic scenes containing a small number of short audio events. It remains difficult to transform a single written caption into a complex audio sequence with long-term structures and large numbers of stems, such as music clips.

In order to attain high fidelity and long-term coherence across seconds, AudioLM[2] approaches audio synthesis as a speech modeling task in a discrete representation space employing a hierarchy of coarse to fine discrete audio units or tokens. Additionally, since no presumptions are made regarding the content of the audio stream, AudioLM learns how to produce realistic sounds from a pure audio corpus without annotations. Such systems could possibly produce more insightful answers if they are trained on the appropriate data and have the ability to model a range of signals.

The lack of linked speech-to-text data is a barrier to progress and exacerbates the challenges inherent in high-quality, consistent speech synthesis. Compare this to the imaging field. Recent advances in superior image generation quality have greatly contributed to the utilization of large datasets. Additionally, writing a description of a common sound in words is much more difficult than writing a description of a graphic in text. First, it is difficult to clearly define the essential elements of the acoustic environment and music in one word. Because the audio is organized along a timeline, the captions for the entire series contain far fewer annotations than the captions.

The MusicLM[1] model converts text descriptions into high-fidelity music. In order to overcome the significant issue of pairwise data scarcity, MuLan is a joint music-text model that has been trained to project music and accompanying text descriptions onto nearby representations in an embedding space. Large audio-only corpora can be trained on using this common embedding space without the need for subtitles. In other words, MuLan embeddings are generated from text input during inference but are computed from audio during conditioning during training. Unlabeled music for very complex text descriptions, such as "a mesmerizing jazz song with an unforgettable saxophone solo and solo singer"

or "90s techno from Berlin with deep bass and powerful kicks" You can train MusicLM to generate them on a huge data set of spacious and harmonious music.

Additionally, the system's ability to process signals other than speech is critical, because some musical qualities may be difficult or impossible to communicate in words. When MuLan embeddings are fed into MusicLM, the sequence of generated tokens differs significantly from the equivalent sequence in the training set. This is due to the fact that MusicLM has been expanded to accept additional melodies in the form of audio as conditioning in order to make music clips that match the intended tune and are generated in accordance with the aesthetics of the provided prompt.

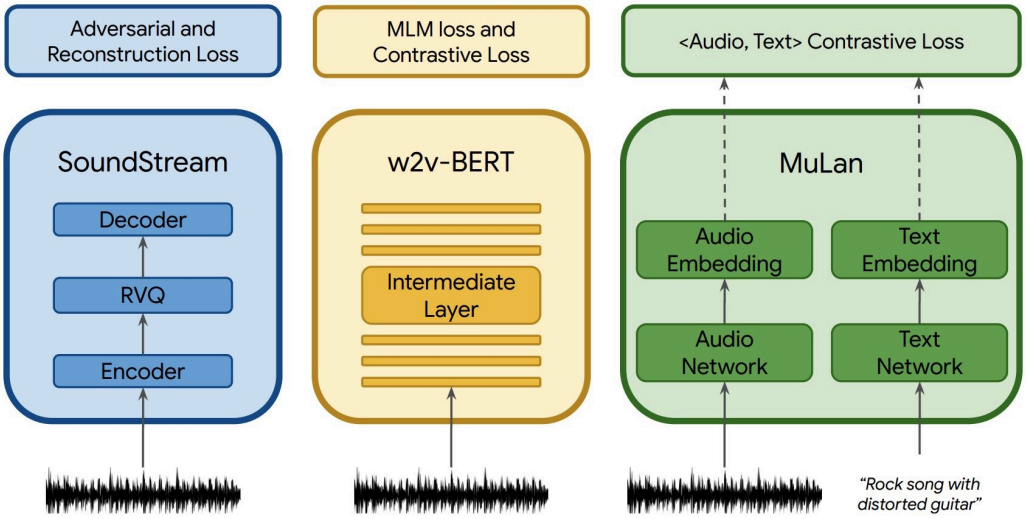


Figure 2.8: Audio and text representations [1]

There are three models that are used for tokenization, which are SoundStream and w2v-BERT as in AudioLM, and MuLan. They're shown in Figure 2.8. Those three models are used for gathering audio representation for conditional autoregressive music generation. The approach in AudioLM also is followed in MusicLM. While w2v-BERT is used for semantic tokens to encourage long-term coherent era, SoundStream is for acoustic tokens, and the MuLan, which

is utilized to symbolize conditioning, inserts content at the time of inference and plays music during training. The discrete sound and text representation for the sequence-to-sequence modeling is made conceivable by these models' autonomous pretrained and frozen nature.

A system that empowers high-quality sound output with consistent structure over long periods of time. This goal consolidates assists propels in adversarial neural sound compression, self-supervised representation learning, and language modeling. Particularly, it begins with raw sound waveforms and creates crude semantic tokens from a pre-trained program on a self-supervised masked language modeling target. In expansion to semantic tokens, the SoundStream neural codec deployment is based on fine-level acoustic tokens that capture subtle elements of the audio waveform and enable high-quality synthesis. Preparing a language model to create both semantic and acoustic tokens at the same time provides high audio quality and long-term consistency.

Additionally, the system's ability to process signals other than speech is critical, because some musical qualities may be difficult or impossible to communicate in words. When MuLan embeddings are fed into MusicLM, the sequence of generated tokens differs significantly from the equivalent sequence in the training set. This is due to the fact that MusicLM has been expanded to accept additional melodies in the form of audio as conditioning in order to make music clips that match the intended tune and are generated in accordance with the aesthetics of the provided prompt.

The SoundStream model is used for mono audio, and RVQ learns 12 quantizers with a vocabulary size of 1024 to quantize the audio embeddings in the training step. The acoustic 600 tokens of A represent a 1-second tone with a bit rate of 6 Kbit/s.Similar to AudioLM, w2v-BERT has 600 parameters to use as an intermediate mask language modeling (MLM) layer. After pre-training and freezing, the seventh layer first collects these embeddings and then places centroids over them and quantizes them using the learned K-means. The target audio representation sequence is obtained from MuLan's audio embedding network. This is a continuous representation that can be used directly as a conditional signal in transformer-based autoregressive models. MuLan embeddings are quantized to provide a discrete representation based on discrete audio and

conditioning signal tokens.

MuLan [15] works with 10 seconds of audio input. So to process longer data, it first processes the long audio input, then it calculates the audio embedding for a 10-second window using steps of 1 second and averages the resulting embedding. The result of this operation is discretized using his RVQ with a vocabulary size of 1024 and a vector quantizer of 12. During prediction, MuLan is conditioned by the text embedding obtained from the text prompt and quantized with the same RVQ used to capture the audio embedding in the 12-token MT. Conditioning during exercise has two benefits. One is that the training data is easy to scale and is not limited by the need for subtitles. Second, it is more responsive to noisy text descriptions.

Hierarchical sequence modeling is used to combine three separate representations to obtain the AudioLM text-based music generation model. The hierarchical model has transformers modeled autoregressively with only a separate decode.MuLan audio tokens, semantic tokens, and acoustic tokens are extracted from the audio-only training set throughout training, and MuLan audio tokens are utilized as conditions for semantic token prediction over the semantic modeling phase. In the semantic modeling stage, the semantic tokens are distributed as $p(S_t|S_{<t}, M_A)$ where S is the semantic tokens and t is the position in sequence according to time step. Acoustic tokens are predicted using MuLan audio and semantic tokens in the subsequent acoustic modeling step. Audio and semantic tokens distributed as $p(A_t|A_{<t}, S, M_A)$ where A_q is the predicted conditioned acoustic tokens. In addition, each stage includes a decoder-only transformer model that is modeled as a sequence-to-sequence task. Finally, the MuLan text token of the conditioning signal computes the text prompt and converts the waveform from the created audio token using the SoundStream decoder.

2.6.5 Textually Guided Audio Generation

Textually Guided Audio generation system AudioGen [19], deals with the sound generation task according to the given text description. It has the same idea as image generation from textual description. Even though image and audio generation have similar approaches, the being one one-dimensional signal for audio is still a challenge from the deep learning perspective. This is a kind of

problem to distinguish overlapping objects. Also not having enough audio data which describes audio as textual is another challenge for this task.

AudioGen is a textually-guided audio generation model that performs text-to-sound generation. It is a single-stage auto-regressive Transformer model trained over a 16kHz EnCodec tokenizer with four codebooks sampled at 50 Hz. Because of the lower frame rate, this model variant achieves similar audio quality to the original approach disclosed in the AudioGen article while delivering faster-generating speed. The AudioGenSolver, which implements the AudioGen's training used to generate the released model, is known as the solver. It defines an autoregressive language modeling task across multiple streams of discrete tokens derived from a pre-trained EnCodec model, similar to MusicGen, with dataset-specific adjustments for environmental sound processing.

This model has been trained with sound data which includes not only musical or melodic sounds but also environmental sounds such as cat meawing, car voice, etc... So when you give a text that describes the environment you want, it produces a sound related to the description keywords included. For instance when you write "A bird is singing while a musician playing guitar in the park with children are playing around, and it starts to rain." In this example, the model needs to generate four different acoustical content but unfortunately, It is quite hard to generate acoustically fidelity sounds because to have the information of degree of positions, decide which composition must be background or foreground in the training set.

AudioGen has 2 steps as in Figure 2.9, first, the raw audio data is encoded to discrete sequences with tokens by a neural audio compression model to train in an end-to-end fashion. Compress representation is reconstructed to raw audio. The second stage works on the output of the first step and also is conditioned textual input. It uses an auto-regressive Transformer-decoder language model to discrete audio tokens. The text is represented by T5 which is a pretrained encoder model on a large corpus of text. Meanwhile, from the given text new set of sounds is sampled with a language model. And those tokens are decoded to waveform for audio representing of output.

For audio representation, it has a similar architecture to the auto-encoder model. The encoder model (E) includes 1D convolutional layers with C channels, followed by B convolutional blocks. Each convolutional block is created with a single residual unit and a down-sampling layer consisting of a stridden convolution with a stride size of S and a kernel size of K. The residual unit comprises two convolutions and a skip connection. The convolutional blocks consist of a two-layer LSTM and end with a 1D convolutional layer with a kernel size of 7 and D output channels.

The flashy is an epoch-based solver module of Facebook that aims to handle 2 tasks such as logging metrics, to multiple back-ends, with custom formatting and check-pointing and automatically tracking stateful parts. Each epoch has several stages for each train, valid, and test dataset. It also provides distributed training.

The aim of training is f jointly minimize a combination of reconstruction losses and adversarial losses. First minimizing the difference between target and regenerated audio data on the time domain and with multiple time scales, calculating and minimizing loss by using the linear combination between two differences $L_1$and L_2 on the mel-spectrogram.

$$\iota_f(x, \widehat{x}) = \frac{1}{|\alpha|.|s|} \sum_{\alpha_i \in \alpha} \sum_{i \in e} \|S_i(x) - S_i(\widehat{x})\|_1 + \alpha_i \|S_i(x) - S_i(\widehat{x})\|_2$$

S_i stands for 64 bins α stands for scalar coefficients

Multi-scale

$$\iota_{feat}(x, \widehat{x}) = \frac{1}{KL} \sum_{K=1} \sum_{L=1} \|D_k(x) - D_k(\widehat{x})\|_1$$

Generation The generation stage enables the generation of samples conditionally and/or unconditionally, as well as audio continuation. It has argmax greedy sampling, softmax sampling at a certain temperature, and top-K and top-P sampling.

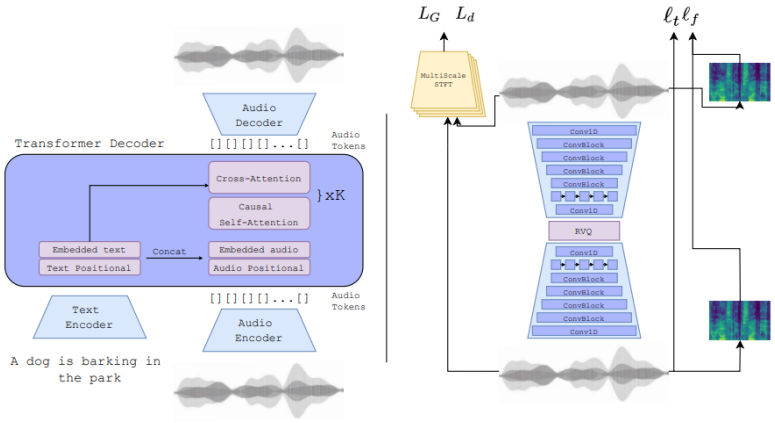


Figure 2.9: AudioGen [19]

2.6.6 Simple and Controllable Music Generation

MusicGen[6] is a text-to-music model capable of classifying high-quality music samples based on linguistic descriptions or audio signals. It is a one-stage autoregressive transformer model trained on a 32 kHz EnCodec tokenizer with four codebooks sampled at 50 Hz, making it a basic and manageable model for creating music[6]. MusicGen, unlike existing methods such as MusicLM, does not require an independent semantic representation and creates all four codebooks at the same time. It demonstrates that it predicts them in parallel with only 50 autoregressive steps per audio second by inserting a slight delay between the codebooks.

It has some challenges which are different from speech generation because of the structure of music. Music has a complex structure because of its features. Besides speech, music has harmonies and melodies from different kinds of instruments. and also full frequency spectrum is needed. Both MusicGen and AudioGen have solvers. Solvers are the core part of these architectures which are the methods of how to solve the given task.

MusicGen solver is called the MusicGenSolver which is responsible for implementing MusicGen's training and defining an autoregressive language modeling task over several streams of discrete tokens taken from a pre-trained EnCodec

model. Of course, the model has several limitations, such as the inability to generate genuine vocals and the fact that it was trained on English descriptions, so it does not function well in other languages. Because there are so many different types of music in the world, it is difficult to train the model with all of them.

T5Tokenizer T5 is an encoder-decoder model that turns all NLP difficulties into a text-to-text format. It is taught by instructor compulsion. This means that during training, we always need an input sequence and a corresponding target sequence. It is a pre-trained language model which is created with encoder and decoder architecture. It is pretrained on both supervised and unsupervised.

The positional embeddings and codebook projections are then given to a transformer-based decoder with L layers, each of which has a causal self-attention and cross-attention block. The conditioning C, which can either be text or melody, is also taken as input by the decoder. After the text has been encoded by a conventional encoder, let's say T5, the cross attention block that accepts the conditioning signal C if is text, says T5. If it is the melody, they pre-process it into a chromatogram and then transmit the conditioning tensor C as a prefix to the transformer input. As a result, we receive the generated output music that is conditioned on condition C. There can be many language models and they can used for the text-to-text part of the model. The models that are also used for the MusicGen project are FLAN-T5 and CLAP. Instead of using T5, also these can be chosen for the training part.

EnCodec

EnCodec is a convolutional-based encoder and decoder for the data. It's a compression model to avoid losing quality. Normally codecs decompose the signal between the frequencies. Most of them refer to the human hearing ability. EnCodec has trained neural networks to regenerate sound from end to end. Because MusicGen is built upon EnCodec's design, the encoder, quantizer, and decoder are the three components that makeup EnCodec's end-to-end architecture. It is similar to the autoencoder module.

While the encoder is for uncompressed data and transforms it to a higher dimensional and lower frame rate to extract latent space, then the quantizer compresses this uncompressed data to the target size. It regenerates the original

signal from the saved information which is the most important one. As the final step of the EnCodec, the decoder converts compressed audio data to time domain space. The converted waveform data is similar to the original one. Identifying changes because of loss is not perceptible to humans, it is the reason that discriminators are used for improving the generated sound's perceptual quality.

A loss balancer method stabilizes training by decoupling hyper-parameter selection from the normal scale of the loss. Furthermore, lightweight Transformer models are employed to condense the resultant representation while retaining real-time speed.The blancer creates a combination of all losses and gradients re-scales. In that way, the model will able to stabilize training,

3 Results

3.1 Audio Generation with XL-Transformers

The transformer architecture for processing was built using the GoodSound dataset. The average audio length is between 4 and 8 seconds. It took too long to make 4 minutes of audio, but the model can produce 40 seconds of audio pretty quickly. The model functions similarly to next word or next sentence prediction. According to the trained data, it continued the sound beyond 4 seconds and created a longer sound from the shorter one.

Waveform data has been converted to MIDI in order to generate Pianoroll data for transformers. After converting the data to Pianoroll, the note information is extracted as words in the NLP task, and the following notes are predicted based on the first 4 seconds of the audio, and the rest of the audio's notes are generated by the model before the notes are converted to an audio file. It gives us the freedom to choose any instrument we desire. Unfortunately, this strategy is insufficient if we wish to make a sound that is unrelated to any instrument that we may play with notes.

These symbolic representations have been addressed through the application of AI for generating cohesive music. However, these perspectives cannot adequately capture the intricacies of emotion and style in music.

3.2 AudioCraft

The audio Craft method has previously been described.As a result, the dataset has been prepared for implementation in accordance with the design of AudioGen and MusicGen. Both models are trained using various hyperparameters. It demonstrates that utilizing a learning scheduler is the most accurate and fast technique to train data. These two models each have their own solver architecture, and the configuration fgilse has been updated to control which techniques are utilized. For both models, multiple experiments were conducted to determine the ideal combination of all optimization, kind of learning schedule, learning rate, batch size, and so on.

Unfortunately, hardware constraints caused us to lower the size of the language models, and batch size. Cross entropy was employed to calculate the loss function. Several learning rate values were tried, and in the end, starting with a higher value of learning rate, increasing the learning for each model with the schedule, and arranging learning rate for smaller values, helped to better configure weights and prevent over-fitting and unbalanced between validation and training.

3.2.1 MusicGen

Hugging Face's model repository is being queried for a pre-trained compression model, especially the "encodec 32khz" model.The compression model in use is configured with four codebooks, each with a cardinality of 2048, implying that each codebook contains 2048 unique entries. Furthermore, this model has a frame-rate of 50, which means it processes data at a rate of 50 frames per second. T5 is a model, and auto-casting is enabled during its evaluation. To maximize efficiency, auto-casting often refers to automatically translating model computations to the float32 (single-precision floating-point) data type.

Exponential Moving Average (EMA) is a technique used in model training. In this scenario, EMA with a decay rate of 0.99 is applied to the model. It means that during training, an exponentially weighted average of the model's parameters with a decay factor of 0.99 is computed, and this procedure is repeated every 30 updates.In terms of parameters, the model has around 21.26 million. This

reflects the model's complexity and capabilities. When the model, gradients, and optimizer are loaded and operational, this indicates the base memory use. The amount of memory used is around 0.34 gigabytes (GB).

Each layer in the model has 256-dimensional hidden representations. It uses eight attention heads for multi-head self-attention, allowing it to capture a wide range of data patterns. The "hidden scale" factor is set to 4, which may improve the model's ability to learn complicated features.

Particularly, this model is intended to be related, which means that it only examines past tokens during self-attention, making it suited for autoregressive model tasks. The Gaussian Error Linear Unit (GELU) is utilized as the activation function, and layer normalization is applied before the feedforward layers.

Changing Learning Scheduler Strategy Unfortunately, without a scheduler strategy, the Adam optimizer fails to reduce cross-entropy. Step Scheduler with a step size of 4000 and a learning rate that starts with 1, allowing for the usage of large learning rates. To avoid over-fitting and improve the variety in the validation set for having a balanced loss value between the training and validation, the training set has 16,000 samples while the validation set has 6000 samples. Otherwise, a lack of samples in the validation set results in a smaller loss for the validation set. This could be because the validation set was too simple for the model. Although D-adaption Adam is not the default optimizer, it works well with the learning scheduler. At the end of the sixth epoch, the model had a loss value of less than 0.5.

Train Summary The last epoch's learning rate is set to 1.00E-04, which controls the size of parameter updates during training. The size of gradients utilized in the training procedure is indicated by the gradient norm of 0.3578. Furthermore, the gradient scale is 1048576.000. The cross-entropy loss, which measures how well the model's predictions match the actual data, is 0.199, while the perplexity, which measures the model's performance in predicting sequences, is 1.221. Finally, the length of this training update is recorded as 209.091 units of time. Figure 3.2 depicts the change in learning rate during the training period.

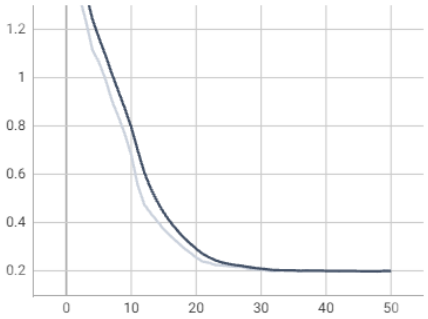


Figure 3.1: MusicGen Training Cross-Entropy

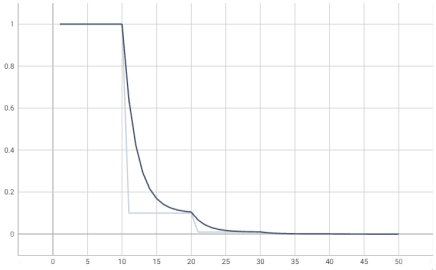


Figure 3.2: MusicGen Learning Rate

Valid Summary Several crucial metrics are included in this summary of the validation procedure for the 50th epoch. The cross-entropy number is 0.197 as shown in the accompanying Figure 3.3. Furthermore, the perplexity is recorded at 1.218. Finally, the duration of this validation task is 106.572 units of time.

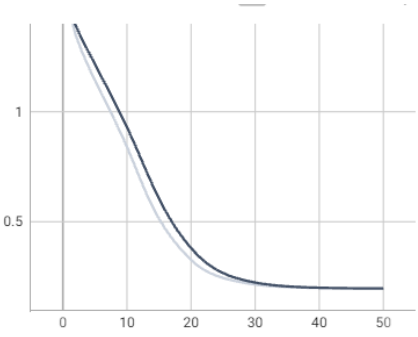


Figure 3.3: MusicGen Validation Cross-Entropy

Evaluate Summary Several critical measures provide insights into the model's performance in the evaluation summary in Epoch 50. The cross-entropy is 0.195, which indicates how closely the model's predictions match the actual data. Perplexity, a common metric in language modeling, is 1.216, showing the model's ability to predict sequences. The chroma cosine similarity, which measures the similarity of chroma features in audio or music analysis, is also recorded at 0.269.

The "rtf" number, which represents the real-time factor, is given as 0.486. This implies that the generating process occurred at around half the pace of real-time, providing insight into its computational efficiency. The duration of this generating work is recorded as 82.791 units of time.

3.2.2 AudioGen

The Hugging Face model hub, specifically the "encodec 32khz" repository, is getting a pre-trained compression model for the model. In this instance, the compression model consists of four codebooks, each having 2048 unique entries (cardinality). Furthermore, the model has a frame-rate of 50 frames per second. The T5 model will be evaluated using auto-casting, which means it will be computed using the float32 (single-precision floating-point) data type. The training method includes decoupled weight decay, which is a technique for independently adjusting the weight decay for distinct regions of the model.

The model is using an EMA with a decay rate of 0.99. During training, this process occurs every 30 updates. By smoothing out the model's parameters, EMA helps to stabilize and improve training. In terms of parameters, the model has around 21.26 million. The model, gradients, and optimizer need approximately 0.34 GB of RAM.

Train Summary The learning rate at epoch 50 is 1.00E-04, the gradient norm is 3.578E-01, the gradient scale is 1048576.000, the cross-entropy is 0.199, the perplexity is 1.221, and the epoch duration is 209.091 units of time. Figure 3.5 depicts the change in learning rate during the training period.

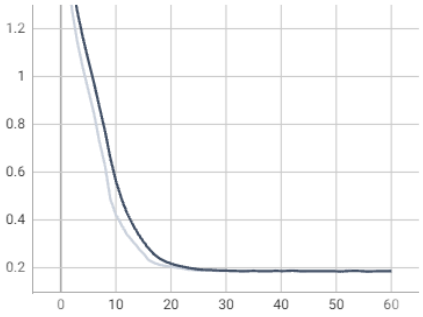


Figure 3.4: AduioGen Training Cross-Entropy

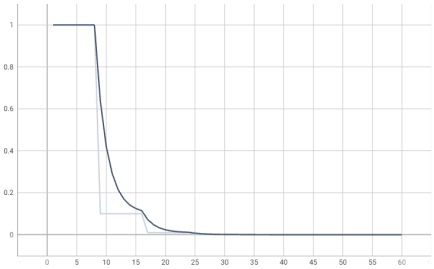


Figure 3.5: AduioGen Learning Rate During Training

Valid Summary As a result, the model attained a cross-entropy of 0.197 and a perplexity of 1.218 at Epoch 50, as shown in the accompanying figure 3.6.

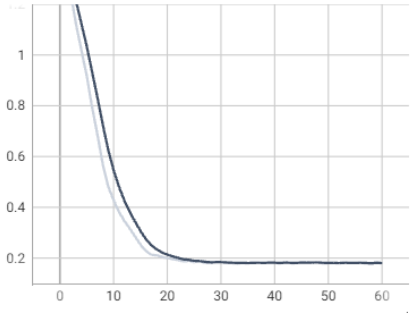
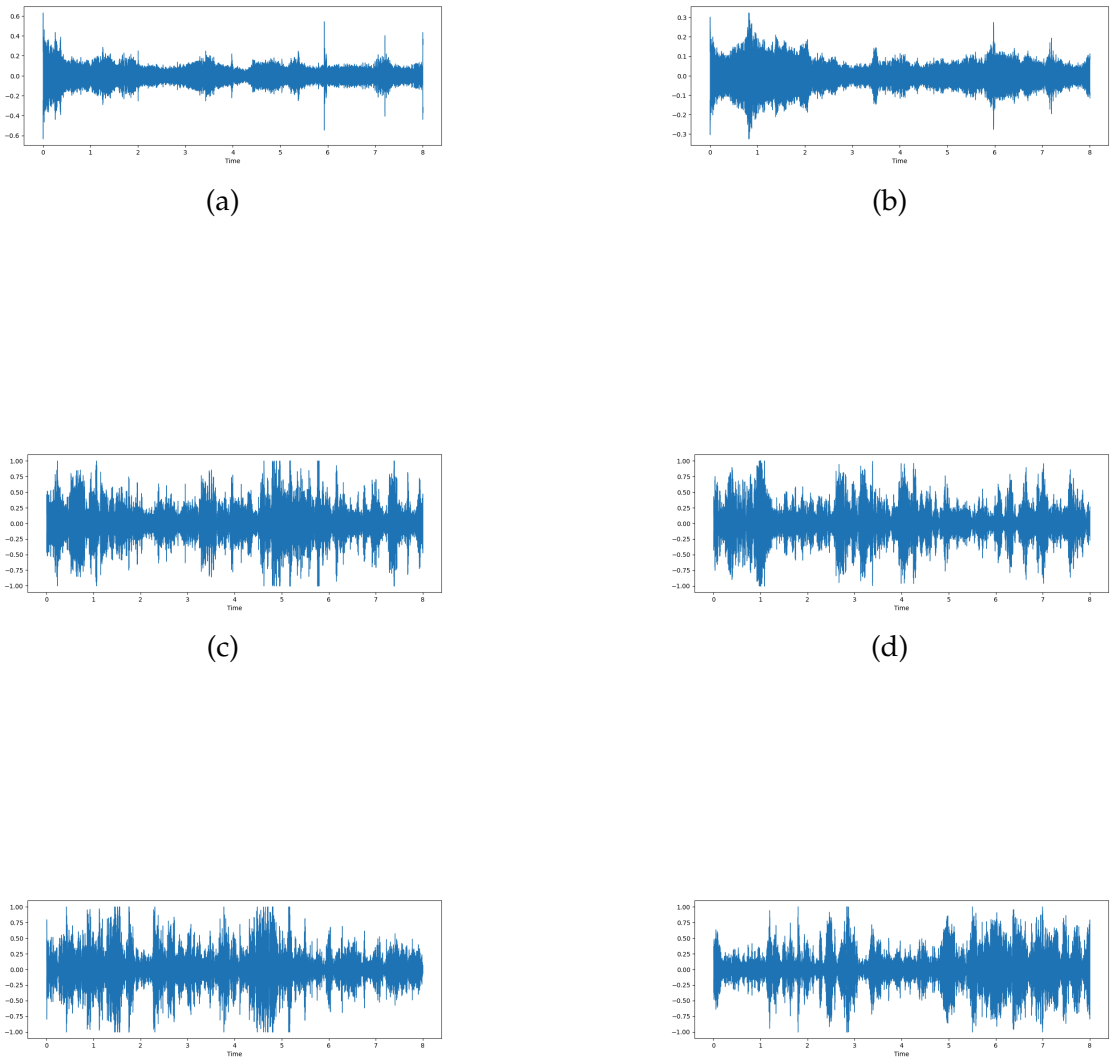


Figure 3.6: AduioGen Valitadion Cross-Entropy

Evaluate Summary In the model, the assessment results show a cross-entropy score of 0.19 and a cosine similarity score of 0.269 after the 50th epoch.

3.2.3 Comparing AudioGen and MusicGen



(a) (b)

(c) (d)

(e) (f)

Figure 3.7: Waveform of MusicGen and AudioGen generated sounds on the left side of the waveforms belong to sounds generated by AudioGen architecture, and on the right-sided ones belong to sounds generated by MusicGen Architecture.

In Figures 3.7, and 3.8, a and b have the same input as text as Continuously Stridulating, and an is the one generated by AudioGen and b is the one generated by MusicGen. Even if they appear to be the same, it is easy to tell them apart. Furthermore, c and e are both a created by AudioGen with only one word difference in the text input, which is synthetic, whereas c lacks the syntetic term but e does. The same is true for d and f in MusicGen.

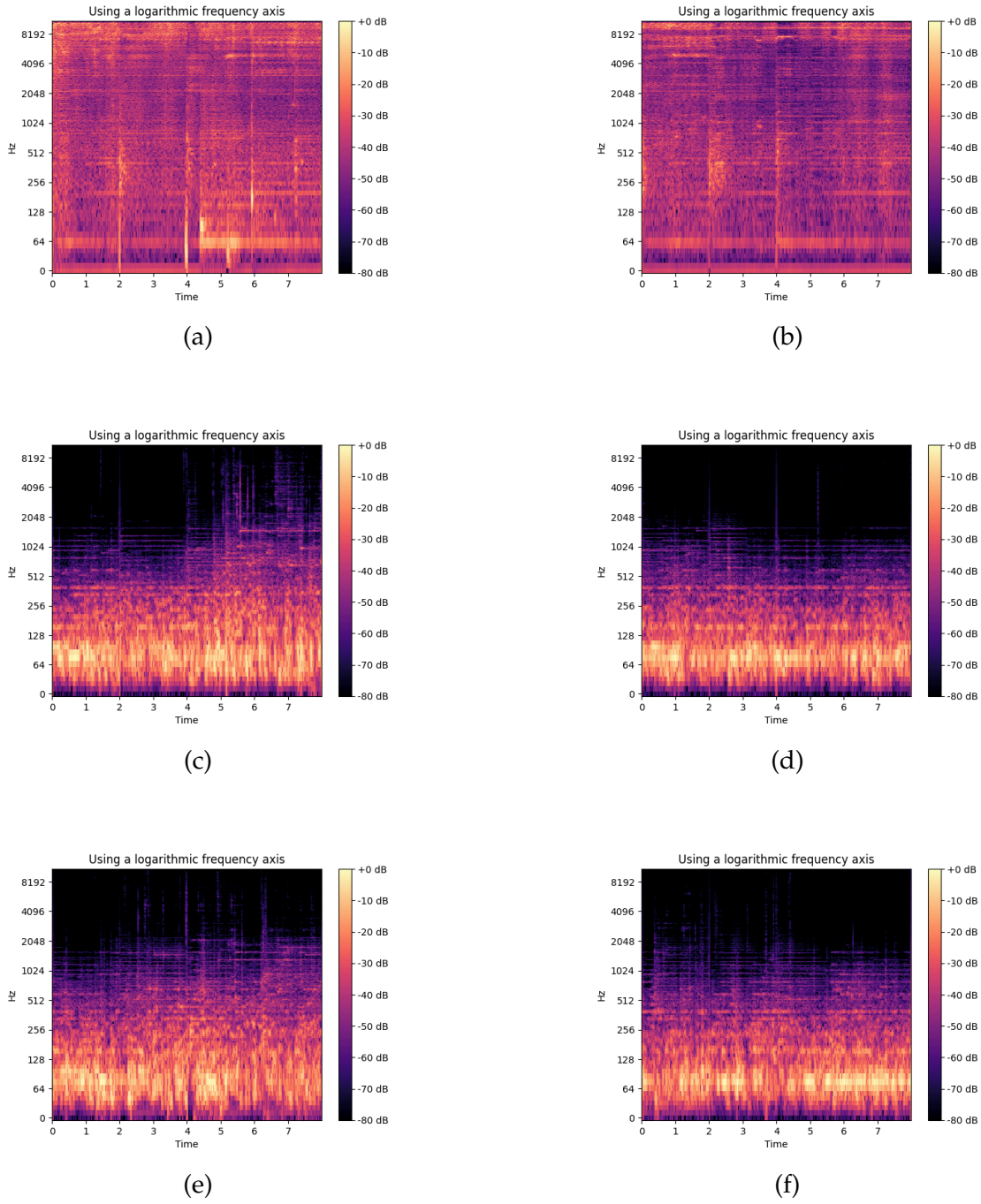


Figure 3.8: Spectrogram with Frequency Axis, on the left side of the spectrograms belong to sounds generated by AudioGen architecture, and on the right-sided ones are belong to sounds generated by MusicGen Architecture.

3.2.4 Text-Sound Match Evaluation

34 people took part in the study's evaluation phase. We did a twofold evaluation in order to better understand the quality and emotional impact of AI-

generated sounds. Participants were shown AI-generated sounds and textual descriptions, and they were asked to judge the accuracy and overall quality of the sounds on a rating scale. This assessment took into account how well the sounds matched the text descriptions provided. Participants investigated the emotional aspects of the sounds. They had to identify and describe the feelings elicited by each sound, as well as indicate the strength of these emotions. Participants also evaluated how well the sounds conveyed the intended emotions as described in the accompanying text descriptions. We gained useful insights into the performance and emotional resonance of the AI-generated sounds as a result of this holistic study.

According to human inspection, both models achieved very close scores of 10 for sound quality. MusicGen gets a rating of 4.93, while AudioGen has a rating of 4.98. All of the sounds generated by AI using both models might have the majority of their labels correct. Figures below show the comprehensive evaluations.

CONTINUOUS DEEP AND SPATIAL STABLE AND RUMBLY

Figure 3.9 depicts the results of this sound, while Table 3.1 summarizes them. The majority of respondents offered responses that correspond to the label descriptions generated by the model. According to the participants, the average quality rating for this sound is 4.2 out of 10.

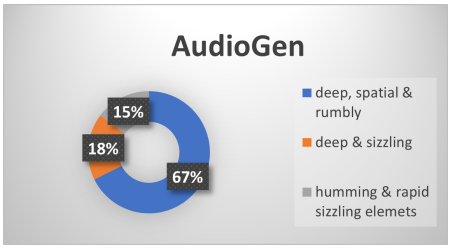


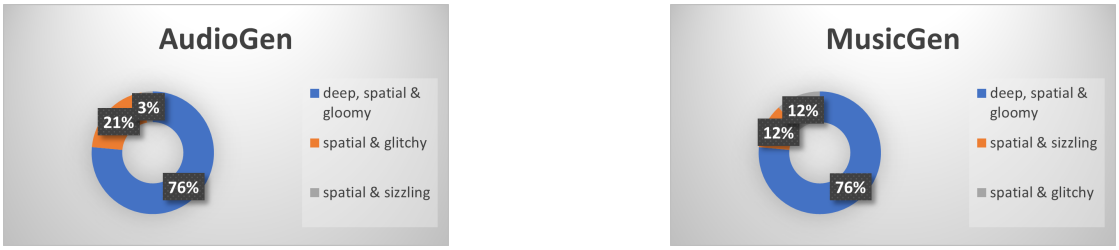
Figure 3.9: Graphic for continuous deep and spatial stable and rumbly

Description	Count
deep, spatial and rumbly	23
deep and sizzling	6
humming and rapid sizzling elemets	5
Grand Total	34

Table 3.1: Answers for continuous deep and spatial stable and rumbly

DEEP, SPATIAL AND GLOOMY

Figure 3.10 depicts the results of this sound, while Table 3.2 and 3.3 summarizes them. The majority of respondents offered responses that correspond to the label descriptions generated by the model. According to the participants, the average quality rating for this sound is 4.5 out of 10 for AudioGen and 4.0 for MusicGen.



(a) (b)

Figure 3.10: Graphic for deep, spatial and gloomy

Description	Count
deep, spatial and gloomy	26
spatial and glitchy	7
spatial and sizzling	1
Grand Total	34

Table 3.2: AudioGen answers for deep, spatial and gloomy

Description	Count
deep, spatial and gloomy	26
spatial and glitchy	4
spatial and sizzling	4
Grand Total	34

Table 3.3: MusicGen answers for deep, spatial and gloomy

This set of responses and graphs suggest that AudioGen is slightly more successful than MusicGen in producing this sound.

DEEP, SPATIAL AND HOLLOW

Figure 3.11 depicts the results of this sound, while Table 3.4 and 3.5 summarizes them. The majority of respondents offered responses that correspond to the label descriptions generated by the model. According to the participants, the average quality rating for this sound is 5.2 out of 10 for AudioGen and 4.6 for MusicGen.

(a) (b)

Figure 3.11: Graphics for deep, spatial, and hollow

Description	Count
deep, spatial, and hollow	24
sizzling and gloomy	6
glitchy and stable	4
Grand Total	34

Table 3.4: AudioGen answers for deep, spatial, and hollow

Description	Count
deep, spatial and hollow	23
sizzling and gloomy	11
Grand Total	34

Table 3.5: MusicGen answers for deep, spatial and hollow

This set of responses and graphs suggest that MusicGen is slightly more successful than AudioGen in producing this sound.

SCARY AND SIZZLING CYMBALS WITH VARYING PITCH

Figure 3.12 depicts the results of this sound, while Table 3.6 and 3.7 summarizes them. The majority of respondents offered responses that correspond to the label descriptions generated by the model. According to the participants, the average quality rating for this sound is 5.3 out of 10 for AudioGen and 4.6 for MusicGen.

(a) (b)

Figure 3.12: Graphics for scary and sizzling cymbals with varying pitch

Description	Count
scary and sizzling cymbals with varying pitch	33
gloomy and humming	1
Grand Total	34

Table 3.6: AudioGen answers for scary and sizzling cymbals with varying pitch

Description	Count
scary and sizzling cymbals with varying pitch	32
stable and rumbly	2
gloomy and humming	1
Grand Total	34

Table 3.7: MusicGen answers for scary and sizzling cymbals with varying pitch

This set of responses and graphs suggest that AudioGen is slightly more successful than MusicGen in producing this sound.

CONTINUOUSLY STRIDULATING BUZZING AND SIZZLING SOUND OF PROCESSED CRICKETS

Figure 3.13 depicts the results of this sound, while Table 3.8 and 3.9 summarizes them. The majority of respondents offered responses that correspond to the label descriptions generated by the model. According to the participants, the average quality rating for this sound is 5.1 out of 10 for AudioGen and 5.1 for MusicGen.

(a) (b)

Figure 3.13: Graphics for by continuously stridulating buzzing and sizzling sound of processed crickets

This set of responses and graphs suggest that MusicGen is slightly more successful than AudioGen in producing this sound.

Description	Count
buzzing, sizzling and sound of crickets	24
science fiction and rumbly	5
gloomy and cymbals with varying pitch	5
Grand Total	34

Table 3.8: AudioGen answers for continuously stridulating buzzing and sizzling sound of processed crickets

Description	Count
buzzing, sizzling, and sound of crickets	26
gloomy and cymbals with varying pitch	6
science fiction and rumbly	2
Grand Total	34

Table 3.9: MusicGen answers for continuously stridulating buzzing and sizzling sound of processed crickets

SCIENCE FICTION WITH A SIXTH INTERVAL AND SLIGHT MOVING PANORAMA

Figure 3.14 depicts the results of this sound, while Table 3.10 and 3.11 summarizes them. The majority of respondents offered responses that correspond to the label descriptions generated by the model. According to the participants, the average quality rating for this sound is 5.8 out of 10 for AudioGen and 5.5 for MusicGen.

(a) (b)

Figure 3.14: Graphics for science fiction with a sixth interval and slight moving panorama

Description	Count
science fiction and slight moving panorama	29
scary sizzling cymbals	3
gloomy and sound of processed crickets	2
Grand Total	34

Table 3.10: MusicGen answers for science fiction with a sixth interval and slight moving panorama

Description	Count
science fiction and slight moving panorama	29
gloomy and sound of processed crickets	3
scary sizzling cymbals	2
Grand Total	34

Table 3.11: MusicGen answers for science fiction with a sixth interval and slight moving panorama

This set of responses and graphs suggest that MusicGen has the same success with AudioGen in producing this sound.

RAPID SIZZLING ELEMENT

Figure 3.15 depicts the results of this sound, while Table 3.12 summarizes them. The majority of respondents offered responses that correspond to the label descriptions generated by the model. According to the participants, the average quality rating for this sound is 4.9 out of 10 for MusicGen.

Description	Count
rapid sizzling	29
spatial and humming	3
rumbly	2
Grand Total	34

Table 3.12: MusicGen answers for Rapid sizzling element

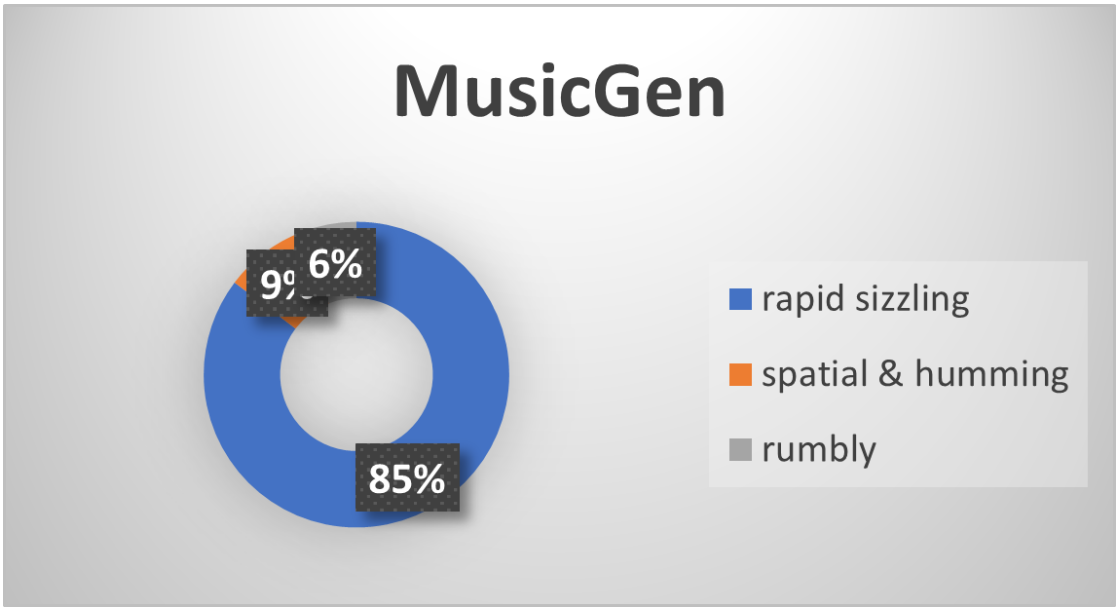
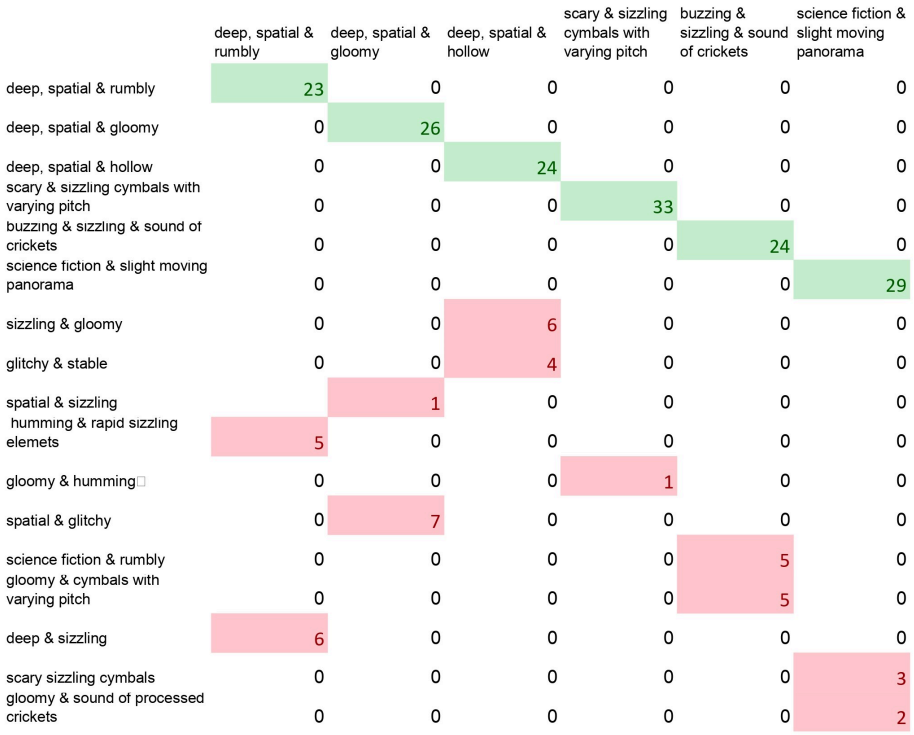


Figure 3.15: Graphic for rapid sizzling element

In summary, the two language models MusicGen and AudioGen were trained to make equivalent sounds using 5500 noise sounds such as humming, sizzling, and so on. These kinds of auditory emotional effects on people were then investigated. According to the participants, there were no discernible changes in the results of the two algorithms for the assigning and qualification parts. To a similar extent, both models accurately correlated written cues with sound quality ratings. The majority of participants correctly identified the textual cue for each sound. However, no perfect 100% connection in their responses to any sound was detected.

The labels provided by the 34 participants closely aligned with the actual text instructions for the associated noises, as seen in Figures 3.16 and 3.17. This observation implies that humans may easily give labels to varied noises or sounds. Participants were able to recognize and classify noises, however, it was note worthy that they did not perceive these sounds as of high quality. Furthermore, in another aspect of the study, the results show that the language models AudioGen and MusicGen generate successful sounds depending on the supplied text cues. It is simple to state that both the AI and human sides validate each other.



	deep, spatial & rumbly	deep, spatial & gloomy	deep, spatial & hollow	scary & sizzling cymbals with varying pitch	buzzing & sizzling & sound of crickets	science fiction & slight moving panorama
deep, spatial & rumbly	23	0	0	0	0	0
deep, spatial & gloomy	0	26	0	0	0	0
deep, spatial & hollow	0	0	24	0	0	0
scary & sizzling cymbals with varying pitch	0	0	0	33	0	0
buzzing & sizzling & sound of crickets	0	0	0	0	24	0
science fiction & slight moving panorama	0	0	0	0	0	29
sizzling & gloomy	0	0	6	0	0	0
glitchy & stable	0	0	4	0	0	0
spatial & sizzling		1	0	0	0	0
humming & rapid sizzling elemets	5	0	0	0	0	0
gloomy & humming	0	0	0	1	0	0
spatial & glitchy	0	7	0	0	0	0
science fiction & rumbly	0	0	0	0	5	0
gloomy & cymbals with varying pitch	0	0	0	0	5	0
deep & sizzling	6	0	0	0	0	0
scary sizzling cymbals	0	0	0	0	0	3
gloomy & sound of processed crickets	0	0	0	0	0	2

Figure 3.16: AudioGen Confusion Matrix

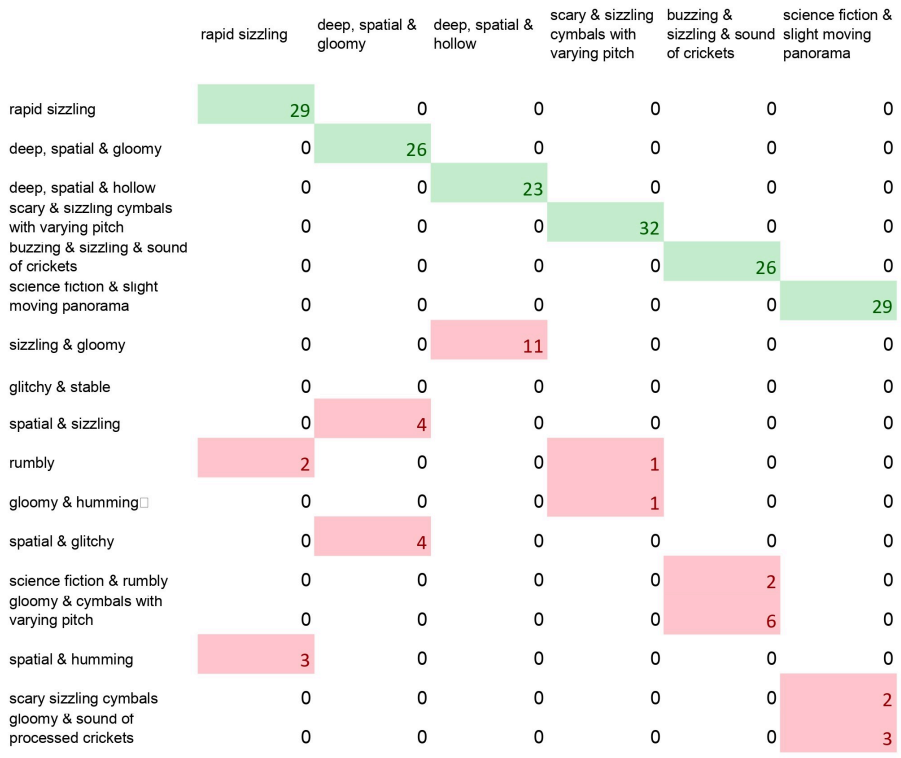


Figure 3.17: MusicGen Confusion Matrix

Emotional Evaluation

The experiment included a group of 34 people who were each given a task. While listening to different noises, participants were asked to describe the strength or amount to which they felt four specific sentiments. These emotions could include things like happiness, sadness, fear, or anger.

The participants were asked to rate how much they felt each of these emotions (happiness, sadness, fear, and anger) on a scale of 0 to 10. This scale allows participants to express the intensity or degree of their emotions while listening to the noises.

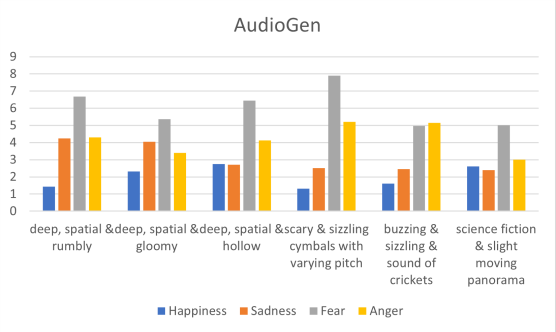


Figure 3.18: AudioGen the emotions conveyed by the sounds

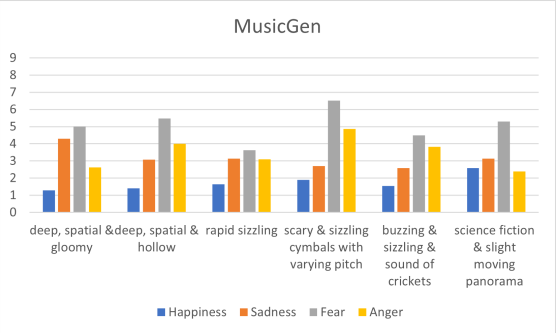


Figure 3.19: MusicGen the emotions conveyed by the sounds

When we inspect Figures 3.11 and 3.12, we see that both models provide the correct noises with high accuracy. Based on this data, it is possible to conclude that these sounds elicit fear followed by anger in individuals. People become more apprehensive when exposed to these sounds. Furthermore, it is clear that happiness is represented at a lesser percentage.

Also, since the sound contains "scary" as a description within, it is easy to say that it is the one that makes the participants feel more fearful than the other noises made. As a result, we can easily say that we may express emotions in the description to make people feel that emotion. It enables us to alter their emotions by using noises generated by descriptions.

Based on the results of two different models, it appears that the noise generated by AudioGen had a greater influence on the participants than the noise generated by MusicGen.

4

Conclusions and Future Works

To summarize, audio and music creation are distinct tasks. To achieve realistic results, many models must be trained. There are difficulties since music has a broad scope. Music generation is more specialized than audio generation, however, there are some difficulties to address. The generation is also affected by the data collection and the type of audio output sought. To improve the quality of encoder-decoder models, but to produce longer-created sounds from short-length sounds, language models, particularly transformers, are extremely efficient and coherent.

Longer outputs, higher sound quality, acoustical position, coherence sounds, and other factors must be considered separately, and sound generating requires more than one model to address those factors. As a result, combining the answers to various problems allows for better results. With only one model to generate sound, if the dataset is of poor quality, the outputs will be of poor quality as well due to the poor quality of the reference data, hence those types of model outputs require pre-processing and post-processing to improve the sound quality. Because most models can create the same length as in the training dataset, the length must be addressed as a job when adding some feature for continuous generation.

Meanwhile, if the task needs to be broadened, such as text-to-sound or text-to-music models, more models must be trained, not only to generate sound but also to grasp the text semantically. Text-to-music and text-to-sound models are

quite beneficial for obtaining desired sounds for the food sound project. Text-to-music is a smaller version of text-to-sound. Because the task is as broad as audio production, because it requires more descriptions and sounds to build tokens and matches, text-to-music would suffice for this purpose in terms of decreasing model parameters.

At this point, transformers by themselves are very promising for longer generations due to audio representation, and it has some limitations such as being good for instrument generation. Symbolic representations are preferable for this paradigm due to its architecture and language approach. It considers the task to be a language processing task, such as next-word prediction or machine translation. As a result, RNNs, LSTMs, and transformers are all effective models for the textual representation of music.

We also asked people to fill out a survey about how they felt after hearing the noises we created. So, for the model text-to-music or text-to-sound, feelings can be included in the training dataset by employing this type of information. So they can be generated by feeling as well as by describing what kind of noises they are. We were able to collect data on the sounds we created as well as how people felt while listening.

For the following works, sound can be made using these emotions and possibly in the future sense based on people's experiences with their emotions and senses. This model can now make audio with descriptions, but based on the information obtained from individuals, it can also make sounds with emotions and later reference senses that need to affect people's perceptions.

References

[1] Andrea Agostinelli et al. "Musiclm: Generating music from text". In: *arXiv preprint arXiv:2301.11325* (2023).

[2] Zalán Borsos et al. "Audiolm: a language modeling approach to audio generation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2023).

[3] Gino Brunner et al. "Symbolic Music Genre Transfer with CycleGAN". In: *Computing Research Repository (CoRR)* abs/1809.07575 (2018). arXiv: 1809.07575. URL: http://arxiv.org/abs/1809.07575.

[4] Yu-Hua Chen et al. "Automatic Composition of Guitar Tabs by Transformers and Groove Modeling". In: *CoRR* abs/2008.01431 (2020). arXiv: 2008.01431. URL: https://arxiv.org/abs/2008.01431.

[5] Ondrej Cfka et al. "Self-Supervised VQ-VAE For One-Shot Music Style Transfer". In: *CoRR* abs/2102.05749 (2021). arXiv: 2102.05749. URL: https://arxiv.org/abs/2102.05749.

[6] Jade Copet et al. *Simple and Controllable Music Generation*. 2023. arXiv: 2306.05284 [cs.SD].

[7] Zihang Dai et al. "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context". In: *CoRR* abs/1901.02860 (2019). arXiv: 1901.02860. URL: http://arxiv.org/abs/1901.02860.

[8] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[9] Sander Dieleman, Aäron van den Oord, and Karen Simonyan. "The challenge of realistic music generation: modelling raw audio at scale". In: *CoRR* abs/1806.10474 (2018). arXiv: 1806.10474. URL: http://arxiv.org/abs/1806.10474.

[10] Chris Donahue, Julian McAuley, and Miller Puckette. *Adversarial Audio Synthesis*. 2019. arXiv: 1802.04208 [cs.SD].

[11] Jesse H. Engel et al. "GANSynth: Adversarial Neural Audio Synthesis". In: *Computing Research Repository (CoRR)* abs/1902.08710 (2019). arXiv: 1902.08710. URL: http://arxiv.org/abs/1902.08710.

[12] Jesse H. Engel et al. "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders". In: *Computing Research Repository (CoRR)* abs/1704.01279 (2017). arXiv: 1704.01279. URL: http://arxiv.org/abs/1704.01279.

[13] Josh Gardner et al. "MT3: Multi-Task Multitrack Music Transcription". In: *CoRR* abs/2111.03017 (2021). arXiv: 2111.03017. URL: https://arxiv.org/abs/2111.03017.

[14] Sangjun Han, Hyeongrae Ihm, and Woohyung Lim. "Symbolic Music Loop Generation with VQ-VAE". In: *CoRR* abs/2111.07657 (2021). arXiv: 2111.07657. URL: https://arxiv.org/abs/2111.07657.

[15] Qingqing Huang et al. "MuLan: A Joint Embedding of Music Audio and Natural Language". In: *International Society for Music Information Retrieval Conference*. 2022. URL: https://api.semanticscholar.org/CorpusID:251881345.

[16] Yu-Siang Huang and Yi-Hsuan Yang. "Pop Music Transformer: Generating Music with Rhythm and Harmony". In: *CoRR* abs/2002.00212 (2020). arXiv: 2002.00212. URL: https://arxiv.org/abs/2002.00212.

[17] Dasaem Jeong, Taegyun Kwon, and Juhan Nam. "VirtuosoNet : A Hierarchical Attention RNN for Generating Expressive Piano Performance from Music Score". In: 2018. URL: https://api.semanticscholar.org/CorpusID:201787400.

[18] Shulei Ji, Jing Luo, and Xinyu Yang. "A Comprehensive Survey on Deep Music Generation: Multi-level Representations, Algorithms, Evaluations, and Future Directions". In: *Computing Research Repository (CoRR)* abs/2011.06801 (2020). arXiv: 2011.06801. URL: https://arxiv.org/abs/2011.06801.

[19] Felix Kreuk et al. "Audiogen: Textually guided audio generation". In: *arXiv preprint arXiv:2209.15352* (2022).

[20] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *CoRR* abs/1411.1784 (2014). arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.

[21] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. "Neural Discrete Representation Learning". In: *Computing Research Repository (CoRR)* abs/1711.00937 (2017). arXiv: 1711.00937. URL: http://arxiv.org/abs/1711.00937.

[22] Aäron van den Oord et al. "WaveNet: A Generative Model for Raw Audio". In: *Computing Research Repository (CoRR)* abs/1609.03499 (2016). arXiv: 1609.03499. URL: http://arxiv.org/abs/1609.03499.

[23] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[24] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. "Self-Attention with Relative Position Representations". In: *CoRR* abs/1803.02155 (2018). arXiv: 1803.02155. URL: http://arxiv.org/abs/1803.02155.

[25] Ian Simon and Sageev Oore. *Performance RNN: Generating Music with Expressive Timing and Dynamics*. https://magenta.tensorflow.org/performance-rnn. Blog. 2017.

[26] Charles Spence and Nicoletta Di Stefano. "Coloured Hearing, Colour Music, Colour Organs, and the Search for Perceptually Meaningful Correspondences between Colour and Sound". In: *i-Perception* 13.3 (2022), p. 20416695221092802. DOI: 10.1177/20416695221092802.

[27] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215 (2014). arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

[28] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[29] Prateek Verma and Jonathan Berger. "Audio Transformers: Transformer Architectures For Large Scale Audio Understanding. Adieu Convolutions". In: *CoRR* abs/2105.00335 (2021). arXiv: 2105.00335. URL: https://arxiv.org/abs/2105.00335.

[30] Prateek Verma and Chris Chafe. "A Generative Model for Raw Audio Using Transformer Architectures". In: *CoRR* abs/2106.16036 (2021). arXiv: 2106.16036. URL: https://arxiv.org/abs/2106.16036.

[31] Jian Wu et al. “A Hierarchical Recurrent Neural Network for Symbolic Melody Generation”. In: *CoRR* abs/1712.05274 (2017). arXiv: 1712.05274. URL: http://arxiv.org/abs/1712.05274.

[32] MASSIMILIANO ZAMPINI and CHARLES SPENCE. “THE ROLE OF AUDITORY CUES IN MODULATING THE PERCEIVED CRISPNESS AND STALENESS OF POTATO CHIPS”. In: *Journal of Sensory Studies* 19.5 (), pp. 347–363. DOI: https://doi.org/10.1111/j.1745-459x.2004.080403.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1745-459x.2004.080403.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1745-459x.2004.080403.x.

[33] Jing-Xuan Zhang et al. “Sequence-to-Sequence Acoustic Modeling for Voice Conversion”. In: *CoRR* abs/1810.06865 (2018). arXiv: 1810.06865. URL: http://arxiv.org/abs/1810.06865.

Acknowledgments

I would like to express my gratitude to Prof. Antonio Rodà, my esteemed advisor, for all the guidance, support, and instruction he provided me throughout my master's thesis studies. I would like to thank the Department of Information Engineering at the University of Padua for providing me with the resources to pursue graduate study.

In addition, I would like to thank Filippo Carnovalini and Alessandro Fiordelmondo, whose invaluable feedback greatly influenced how I conducted my experiments and interpreted my findings. Friends, lab mates, colleagues, and research team Chiara De Luca, Edoardo Bastianello, Gauri Pravishiare all appreciated for the fun times we had working and socializing together. I would also like to thank everyone who has been there for me emotionally and intellectually as I have worked on my coursework.