GPT – DRIVEN SONG GENERATION: EXPLORING AUTOMATED MUSIC CREATION WITH AI

PROJECT REPORT

21AD1513- INNOVATION PRACTICES LAB

Submitted by

CHERLIN FLORY THOMAS (211422243051)

ANU SUSHMITHA. S (211422243024)

MAITHREAYI. P (211422243183)

in partial fulfillment of the requirements for the award of degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123

ANNA UNIVERSITY: CHENNAI-600 025

October, 2024

BONAFIDE CERTIFICATE

Certified that this project report titled "**GPT – DRIVEN SONG GENERATION: EXPLORING AUTOMATED MUSIC CREATION WITH AI**" is the Bonafide work of **CHERLIN FLORY THOMAS**, **ANU SUSHMITHA. S, MAITHREAYI. P**, Register No. **(211422243051), (211422243024), (211422243183)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

INTERNAL GUIDE
Mrs.V.REKHA, M.E
Assistant Professor
Department of AI &DS

HEAD OF THE DEPARTMENT
Dr.S.MALATHI M.E., Ph.D
Professor and Head,
Department of AI & DS.

Certified that the candidate was examined in the Viva-Voce Examination held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

We introduce SongGen, an advanced language model (LLM) specifically designed for song composition. This model is capable of understanding and generating both melodies and lyrics in symbolic song formats by leveraging the computational capabilities of LLMs. Unlike current music-related LLMs, which process music as quantized audio signals—resulting in inefficient encoding and reduced flexibility—our approach adopts symbolic song representation, a structured and efficient method traditionally used in music composition. This allows the model to compose songs with greater precision and clarity. We propose a novel tuple format that integrates lyrics with three key musical attributes—pitch, duration, and rest duration—ensuring accurate interpretation of musical symbols and precise alignment between lyrics and melody. To provide the model with fundamental musical knowledge, we developed SongGen-PT, a large-scale pretraining dataset comprising lyrics, melodies, and lyric-melody pairs in English. Following extensive pretraining, the model was further refined with 10,000 carefully crafted question-answer pairs, equipping it with robust instruction-following abilities to address a wide range of tasks. Through rigorous evaluation, SongGen demonstrates superior performance in tasks such as lyric-to-melody generation, melody-to-lyric generation, song continuation, and text-to-song creation, consistently outperforming state-of-the-art models like GPT-4.

Keywords: Generative Pre-trained Transformer (GPT), Automated Music Creation, Natural Language Processing (NLP), Neural Networks, MIDI Processing

ACKNOWLEDGEMENT

I also take this opportunity to thank all the Faculty and Non-Teaching Staff Members of Department of Artificial Intelligence and Data Science for their constant support. Finally, I thank each and every one who helped me to complete this project. At the outset we would like to express our gratitude to our beloved respected Chairman, **Dr.Jeppiaar M.A.,Ph.D,** Our beloved correspondent and Secretary **Mr.P.Chinnadurai M.A., M.Phil., Ph.D.,** and our esteemed director for their support.

We would like to express thanks to our Principal, **Dr. K. Mani M.E., Ph.D.,** for having extended his guidance and cooperation.

We would also like to thank our Head of the Department, **Dr.S.Malathi M,E.,Ph.D.,** of Artificial Intelligence and Data Science for her encouragement.

Personally, we thank **Mrs.V.REKHA, M.E.,** Assistant Professor, Department of Artificial Intelligence and Data Science for the persistent motivation and support for this project, who at all times was the mentor of germination of the project from a small idea.

We express our thanks to the project coordinator **Ms.K.CHARULATHA M.E.,** Assistant Professor in Department of Artificial Intelligence and Data Science for their Valuable suggestions from time to time at every stage of our project.

Finally, we would like to take this opportunity to thank our family members, friends, and well-wishers who have helped us for the successful completion of our project.

We also take the opportunity to thank all faculty and non-teaching staff members in our department for their timely guidance in completing our project.

CHERLIN FLORY THOMAS

ANU SUSHMITHA. S

MAITHREAYI. P

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	**3**
	LIST OF FIGURES	**10**
	LIST OF ABBREVIATIONS	**11**
1	**INTRODUCTION** 1.1 Background and Motivation 1.2 Purpose of GPT in Songwriting 1.3 Structure of the Report and Understanding GPT Technology 1.4 Architecture Diagram	**12**
2	**LITERATURE REVIEW** 2.1 Few-Shot Learning with Large Language Models 2.2 Bidirectional Encoder Representations from Transformers (BERT) 2.3 Pathways for Large-Scale Language Modeling 2.4 Music Generation with Diffusion Models 2.5 Dynamic Time Warping for Lyric-Melody Alignment 2.6 Text2vec Toolkit for Evaluating Text Generation	**20**

3	**RELATED WORK** 3.1 MusicLM: Generating Music from Text 3.2 MusiCoder: A Symbolic Music Composition Framework Using Transformers 3.3 POP909: A Pop-song Dataset for Music Arrangement Generation 3.4 Text2Melody: Transformer-Based Generation of Melodic Sequences from Textual Descriptions 3.5 Symbolic Music Generation with Transformer GANs	**23**
4	**PROPOSED WORK** 4.1 Platform Architecture 4.1.1 Data Collection and Preprocessing Layer 4.1.2 Data Preparation Layer 4.1.3 Model Training Layer 4.1.4 Model Components Layer 4.1.5 Song Generation Tasks Layer 4.2 Key functionalities 4.2.1 Text-to-Song Generation 4.2.2 Melody-Based Lyric Generation 4.3 Approach 4.3.1 Model Training and Fine-Tuning	**26**

	4.3.2 Prompt Engineering for Composition Guidance 4.3.3 Integration of Musical Elements	
5	**MODULES** 5.1 Algorithmic Components: Symbolic Song Representation & Tuple-based Data Formatting 5.2 Data Handling and Processing Units: Pretraining Dataset 5.3 Training and Learning Procedures: Two-Stage Training Paradigm 5.4 Specialized Tools: Music Source Separation & Dynamic Time Warping	**32**
6	**SYSTEM REQUIREMENT** 6.1 Introduction 6.2 Software requirement 6.3 Hardware requirement	**38**
7	**CODE**	**42**
8	**OUTPUT**	**51**
9	**WORK AND IMPLEMENTATION** 9.1 Symbolic Representation for LLM 9.1.1 Tuple Data Format 9.1.2 Discretized Duration 9.1.3 Vocabulary Extension 9.2 Two-Stage Training Paradigm	**53**

	9.2.1 Pretraining Stage 9.2.2 Supervised Fine-Tuning Stage	
10	**CONCLUSION AND FUTUREWORK** 10.1 Conclusion 10.2 Future Works	**57**
11	**REFERENCES**	**61**

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO
1	Architecture diagram of a GPT – Driven Song Generation: Exploring Automated Music Creation with AI	16

LIST OF ABBREVIATIONS

AI – Artificial Intelligence
NLP – Natural Language Processing
MIDI – Musical Instrument Digital Interface
BERT – Bidirectional Encoder Representations from Transformers
DTW – Dynamic Time Warping
API – Application Programming Interface
GAN – Generative Adversarial Network
UVR – Ultimate Vocal Remover
LLM – Large Language Model
QA – Question Answering
GPT – Generative Pre-trained Transformer
JSON – JavaScript Object Notation

CHAPTER 1

INTRODUCTION

CHAPTER 1

1. INTRODUCTION

The rise of artificial intelligence (AI) in creative domains has opened new possibilities for automated music creation, particularly through powerful language models like GPT (Generative Pre-trained Transformer). As AI continues to advance, its applications in music are pushing the boundaries of creativity, allowing for the generation of not only lyrics but also melodies, harmonies, and genres. GPT-driven song generation represents a unique intersection of technology and artistry, where AI aids in crafting songs with thematic coherence, emotional depth, and stylistic diversity.

AI-driven songwriting systems leverage vast datasets and complex algorithms to mimic the subtleties of human expression while adapting to various musical styles and emotional tones. This innovation addresses both the high demand for music in commercial media and the individual pursuit of creative expression, democratizing music production and making it accessible to a wider audience. However, the process presents its own set of challenges: achieving lyrical coherence, emotional relevance, and an authentic "human" touch remain key areas for ongoing development.

This research delves into the background, technological underpinnings, techniques, and practical applications of GPT-driven music production. It dives into the training methods, obstacles, and ethical issues, as well as the transformational potential of this technology in the music business and other creative disciplines. By evaluating the possibilities and limits of GPT models in songwriting, this study seeks to give a thorough understanding of AI's role in defining the future of music production.

1.1 Background and Motivation

Interest in using machine learning models in artistic disciplines has increased as a result of artificial intelligence's growing capabilities. Particularly, the ability of generative pre-trained transformers (GPT) to produce language that resembles that of a person has aroused interest in their possible uses in songwriting and music production. Writing songs has always been a fundamentally human art form that combines ideas, emotions, and rhythmic patterns to appeal to listeners. But as AI models like GPT evolve, they may be able to work alongside human artists to help with songwriting, style adaptation, and even the creation of coherent storylines inside songs. The potential to simplify music creation, open up new creative avenues, and investigate the limits of artificial intelligence's contribution to artistic expression are what drive this nexus of technology and art.

1.2 Purpose of GPT in Songwriting

The primary purpose of using GPT in songwriting is to augment the creative process by automating parts of lyric generation and exploring musical ideas more efficiently. By leveraging the power of language models, creators can generate lyrics that capture different emotions, themes, and genres, assisting musicians and lyricists with inspiration or even complete lyrical drafts. This serves a dual purpose: it democratizes songwriting by making the process more accessible to aspiring artists and speeds up production for established musicians who may need rapid idea generation. Additionally, AI-driven songwriting can be adapted to various genres and styles, making it a versatile tool that caters to diverse creative needs. As the technology evolves, it aims to enhance human creativity rather than replace it, providing a valuable asset in both commercial music production and personal artistic projects.

1.3 Structure of the Report and Understanding GPT Technology

The purpose of this work is to present a thorough investigation of GPT's function in automated music production. An introduction of GPT technology and its evolution is given first, and then the particular methods for creating song lyrics are examined, including topics like style transfer and emotional tone. The methods used to optimize GPT for music-related tasks, assess its results, and pinpoint the drawbacks and difficulties of AI-driven songwriting are covered in depth in the following sections. The paper also discusses ethical issues and practical uses of AI in creative industries. The findings are discussed in the conclusion, along with potential future developments and multidisciplinary partnerships that might strengthen AI's contribution to music production.

Transformers (GPT) were created to comprehend and produce writing that is similar to that of a person. GPT models, which are based on deep learning principles, can anticipate and produce coherent text sequences after being trained on large datasets. This makes them appropriate for use in creative writing applications, such as songwriting. GPT models may mimic a wide range of literary and creative forms, from informal writing to lyrical expression, since they learn from a vast array of linguistic patterns, structures, and styles. These models are now accurate and flexible, able to produce content that seems artistically sophisticated and contextually relevant thanks to significant developments in natural language processing (NLP), such as transformer architecture and self attention processes. Knowing the basic principles of GPT allows us to understand its advantages and disadvantages in the context of songwriting and music.

1.4 Architecture Diagram

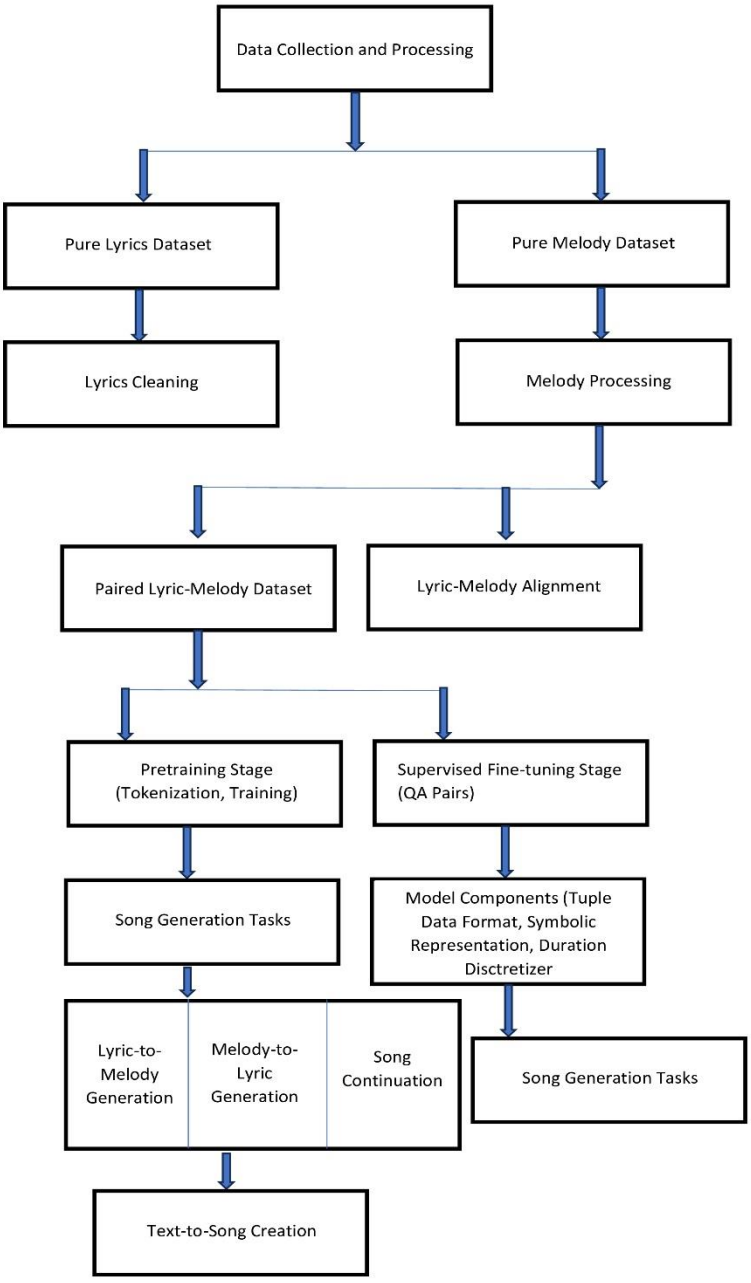


Fig 1.4: Architecture diagram of a GPT – Driven Song Generation: Exploring Automated Music Creation with AI

The architecture of the GPT – Driven Song Generator is composed of five main layers: **Data Collection and Preprocessing Layer**, **Data Preparation Layer, Model Training Layer, Model Components Layer**, and **Song Generation Tasks Layer**. These layers work together to offer seamless song generation.

1. **Data Collection and Preprocessing:**

   Pure Lyrics Dataset: This layer involves gathering a large collection of lyrics in English from various sources. The dataset includes lyrics that are labeled and cleaned for use in training.

   Pure Melody Dataset: This layer involves collecting melody data in the form of MIDI files. These files are parsed to extract musical note attributes like pitch, duration, and rest duration.

   Paired Lyric-Melody Dataset: This involves collecting and synchronizing lyrics with their corresponding melodies. The data is processed to ensure precise alignment at the word level.

2. **Data Preparation**

   Lyrics Cleaning: This sub-layer involves processing the raw lyrics data to remove non-relevant information, such as song titles, artist names, and special symbols, ensuring that only high-quality lyrics are used.

   Melody Processing: This sub-layer parses MIDI files to extract relevant musical attributes and convert them into a structured format suitable for training.

   Lyric-Melody Alignment: This sub-layer uses techniques such as dynamic time warping (DTW) to align lyrics with melody at the word level, ensuring that each word is synchronized with the appropriate notes.

3. **Model Training**

Pretraining Stage:

Data Tokenization: This involves converting the cleaned and processed lyrics and melody data into a tuple format that the model can easily process.

Training on Pure Lyrics and Melody: The model is pretrained using this tokenized data to learn the basic structure and patterns in both lyrics and melody.

Supervised Fine-Tuning Stage:

Instruction-following QA Pairs: The model is further fine-tuned using a set of QA pairs designed to train it on specific tasks such as lyric-to-melody generation, melody-to-lyric generation, song continuation, and text-to-song creation.

4. **Model Components**

Tuple Data Format Handler: This component organizes the input data into a consistent tuple format that represents the relationships between lyrics and melody.

Symbolic Representation Converter: This component converts symbolic song representations into a format that the model can process effectively, preserving the musical context.

Duration Discretizer: This component encodes and decodes note durations into discrete values, allowing the model to handle the timing aspects of the music accurately.

Vocabulary Extension Module: This component extends the model's vocabulary to include specific tokens for musical notes and durations, enabling it to understand and generate musical content.

5. **Song Generation Tasks**

   Lyric-to-Melody Generation: This task involves generating a melody that fits given lyrics. The model uses its understanding of lyrical structure to create a harmonious melody.

   Melody-to-Lyric Generation: This task involves generating lyrics that fit a given melody. The model ensures that the generated lyrics are both meaningful and musically coherent.

   Song Continuation: This task involves extending a given song segment both melodically and lyrically, maintaining the style and coherence of the original piece.

   Text-to-Song Creation: This task involves generating a complete song from a textual description, capturing the themes and emotions described in the text and translating them into both lyrics and melody.

CHAPTER 2

LITRATURE REVIEW

CHAPTER 2

2. LITRATURE REVIEW

The integration of AI in music composition has gained significant traction over recent years, with multiple models and platforms demonstrating the potential of automated music creation. Various systems, such as OpenAI's MuseNet, Jukedeck, Amper Music, AIVA (Artificial Intelligence Virtual Artist), and DeepBach, have showcased different approaches and capabilities in this domain, each with distinct features and limitations. Below are key contributions from the literature that shape the foundation of this platform:

2.1 Few-Shot Learning with Large Language Models

Brown et al. (2020)

This paper introduces the concept of few-shot learning using large language models, which allows SongGen to generate songs with minimal training examples by understanding and adapting to new tasks efficiently. Leveraging large language models for few-shot learning, enabling the model to perform tasks with minimal examples.

2.2 Bidirectional Encoder Representations from Transformers (BERT)

Devlin et al. (2018)

BERT's approach to pre-training a model on large text corpora to understand context and relationships between words is adapted in SongGen for understanding and generating lyrics. Pre-training a deep bidirectional transformer model for understanding contextual relations in language.

2.3 Pathways for Large-Scale Language Modeling

Chowdhery et al. (2023)

The Pathways system enables the scaling of language models to handle extensive datasets and diverse tasks, which is crucial for training SongGen on a large corpus of lyrics and melodies. Scaling language models using the Pathways system to handle large and diverse data efficiently.

2.4 Music Generation with Diffusion Models

Schneider et al. (2023)

This paper's approach of using diffusion models to generate music from text influences SongGen's ability to convert textual prompts into coherent melodies. Using diffusion models for generating music from textual descriptions.

2.5 Dynamic Time Warping for Lyric-Melody Alignment

Müller (2007)

DTW is used in SongGen to ensure precise synchronization between lyrics and their corresponding melody, crucial for creating natural and harmonious songs. Using dynamic time warping (DTW) to align lyrics with melody notes accurately.

2.6 Text2vec Toolkit for Evaluating Text Generation

Xu (2023)

The Text2vec toolkit is employed in SongGen to measure the similarity between generated lyrics and original texts, ensuring the quality and coherence of the generated content. Using the Text2vec toolkit for computing cosine similarity and evaluating the quality of generated text.

CHAPTER 3

RELATED WORK

CHAPTER 3

3. RELATED WORK

The development of SongGen, a large language model designed for song composition, builds upon a rich foundation of research in natural language processing, music generation, and machine learning. Here, we highlight several pivotal works that have influenced its creation. These related works encompass advancements in language modeling, innovative techniques for music generation, and the compilation of extensive datasets, each contributing crucial insights and methodologies that enhance the capabilities of SongGen.

3.1 MusicLM: Generating Music from Text

MusicLM is a model that generates music from textual descriptions. It uses a hierarchical sequence-to-sequence model to produce coherent and high-quality music based on detailed textual prompts. The model captures various aspects of music, such as melody, rhythm, and timbre, and aligns them with the textual descriptions provided. Rec's bidirectional nature allows it to predict future items more effectively.

3.2 MusiCoder: A Symbolic Music Composition Framework Using Transformers

MusiCoder leverages transformer models to compose symbolic music. The framework focuses on encoding musical sequences and learning long-term dependencies to generate coherent compositions. It uses a transformer-based architecture to handle the symbolic representation of music, similar to how natural language is processed.

3.3 POP909: A Pop-song Dataset for Music Arrangement Generation

POP909 is a large-scale dataset of pop songs specifically designed for music arrangement generation. The dataset includes symbolic representations of music and detailed annotations, making it suitable for training models to understand and generate music arrangements.

3.4 Text2Melody: Transformer-Based Generation of Melodic Sequences from Textual Descriptions

Text2Melody uses transformers to generate melodic sequences from textual descriptions. The model learns to map descriptive language to corresponding musical elements, creating melodies that reflect the given text. This approach is closely related to SongComposer's text-to-song generation capabilities.

3.5 Symbolic Music Generation with Transformer GANs

This work combines transformers and GANs to generate symbolic music. The model leverages the strengths of both architectures to produce high-quality musical compositions that capture the structure and style of the training data. It focuses on generating music that is both novel and stylistically consistent.

CHAPTER 4

PROPOSED WORK

CHAPTER 4

4. PROPOSED WORK

The proposed work in this report focuses on developing SongGen, an innovative platform for song composition powered by a large language model (LLM). This platform integrates advanced AI models designed for generating lyrics and melodies, leveraging symbolic music representation to ensure efficiency and flexibility. Users will be able to create and customize songs through natural language inputs, with real-time feedback on lyric and melody modifications. The platform's architecture is designed to ensure high responsiveness, user-friendly interaction, and iterative customization capabilities, aimed at enhancing user creativity and satisfaction.

4.1 Platform Architecture

The proposed platform architecture for interactive song composition using generative AI consists of five main components: the Data Collection and Preprocessing Layer, the Data Preparation Layer, the Model Training Layer, the Model Components Layer, and the Song Generation Tasks Layer. These layers enable users to seamlessly compose and customize songs using natural language and symbolic music inputs.

4.1.1 Data Collection and Preprocessing Layer

The Data Collection and Preprocessing Layer involves gathering a comprehensive dataset, named SongCompose-PT, which includes 280K songs of pure lyrics, 20K sets of pure melodies, and 15K paired lyrics and melodies in English. This diverse and extensive dataset forms the foundation for training the AI model. The preprocessing tasks include cleaning the data, removing noise, and standardizing the format to ensure consistency. This layer ensures that the collected data is of high quality and ready for further processing.

4.1.2 Data Preparation Layer

The Data Preparation Layer is responsible for preparing the cleaned data for model training. This involves formatting the data into a novel tuple design that includes lyrics and three note attributes—pitch, duration, and rest duration. The tuples align lyrics with their corresponding melodies, ensuring that the AI model can accurately understand and generate music in a structured and coherent manner. This step also involves augmenting the data through techniques like pitch shifting and duration scaling to enrich the training dataset.

4.1.3 Model Training Layer

The Model Training Layer forms the core of the platform, utilizing advanced AI models, including the large language model specifically trained for song composition. This layer conducts a two-stage training process: extensive pretraining on the SongCompose-PT dataset and supervised fine-tuning with 10K carefully crafted QA pairs to enhance the model's instruction-following capabilities. The training involves optimizing the model to predict the next token based on prior text, thereby learning to generate coherent and stylistically appropriate lyrics and melodies.

4.1.4 Model Components Layer

The Model Components Layer includes the various sub-models and algorithms that constitute the song composition model. This includes components for natural language processing to handle text inputs, symbolic music processing to generate and understand musical notes, and alignment models to ensure coherence between lyrics and melodies. Advanced techniques such as reinforcement learning and transfer learning are integrated to refine the model's performance based on user interactions and feedback.

4.1.5 Song Generation Tasks Layer

The Song Generation Tasks Layer is the user-facing component of the platform, where users interact with the system to create and customize songs. This layer supports various song generation tasks, including:

- Text-to-Song Generation: Users can describe the desired theme, mood, and style of the song in natural language, and the platform generates corresponding lyrics and melodies.
- Melody-Based Lyric Generation: Users can input existing melodies, which the platform analyzes to generate matching lyrics, ensuring coherence and personalization.
- Real-Time Song Manipulation: Users can adjust song features like lyrical content, melody pitch, and rhythm through intuitive controls, with instant feedback on the updates.
- Predefined Song Templates: A library of predefined song templates based on popular music genres and trends is available for users to select as starting points for further customization.

4.2 Key Functionalities

In order to maximize user creativity and engagement, the platform offers the following essential functionalities:

4.2.1 Text-to-Song Generation

The platform can generate complete songs from text inputs, enabling text-to-song customization. By using natural language processing models, users can describe the desired theme, mood, and style of the song in natural language, and the platform generates corresponding lyrics and melodies. The AI adapts to any further textual modifications provided by the user.

4.2.2 Melody-Based Lyric Generation

In Melody-Based Lyric Generation, users can input existing melodies, which the platform's AI algorithms analyze to generate matching lyrics. The extracted features from these melodies, such as rhythm and pitch, are integrated into the lyric generation process, offering a high level of coherence and personalization based on user-provided music.

4.3 Approach

The integration of artificial intelligence in music composition opens new avenues for creativity. This approach involves training models on diverse musical works to generate genre-specific compositions. Through advanced prompt engineering, user inputs are transformed into coherent musical commands that combine essential elements like melody, harmony, and rhythm. A user feedback loop further allows the model to adapt and improve over time, enhancing the music creation experience.

4.3.1 Model Training and Fine-Tuning

Model training begins with pre-training the GPT model on a diverse corpus of music and lyrics, enabling it to understand various musical styles, structures, and thematic elements. Following this, fine-tuning involves adapting the model to specific musical genres, enhancing its ability to generate compositions that align with the distinct characteristics and conventions of each genre.

4.3.2 Prompt Engineering for Composition Guidance

Advanced prompt engineering plays a crucial role in guiding the model's output. By converting genre specifications and contextual hints into precise compositional commands, this technique ensures that the generated music adheres to the desired stylistic elements and compositional techniques, thereby improving the quality and relevance of the output.

4.3.3 Integration of Musical Elements

The model's capabilities extend to the integration of various musical components, such as melody, harmony, rhythm, and optional lyrics. This holistic approach allows for the creation of comprehensive musical outputs that reflect the complexity and richness of real-world compositions, catering to both instrumental and lyrical needs.

CHAPTER 5

MODULES

CHAPTER 5

5. MODULES

In the SongGen project, several core modules work together to facilitate the generation of lyrics and melodies. These modules manage different aspects of data processing, model training, and music generation. Below are four key modules that form the backbone of the project:

5.1 Algorithmic Components: Symbolic Song Representation & Tuple-based Data Formatting

Symbolic Song Representation

The Symbolic Song Representation module addresses the challenge of representing music in a machine-readable format. Unlike traditional approaches that use continuous audio signals, SongGen employs a symbolic representation where musical elements like pitch, duration, and rest durations are encoded as text-based tokens. This is crucial because:

It makes the handling of musical data efficient and allows for better tokenization and model training.

The structure ensures that musical notes and lyrics can be aligned precisely, enabling the model to generate coherent compositions.

This approach also improves flexibility and interpretability, as symbolic representations are easier to manipulate, which is essential for tasks like lyric-to-melody and melody-to-lyric generation.

Tuple-based Data Formatting

The Tuple-based Data Formatting module is used to align and format the input data for the model in a structured way. Each element of a song—whether it's a lyric word, a musical note, or both—is represented in a tuple, which contains all necessary attributes:

For lyrics, the tuple includes the word.

For melody, it includes the pitch, duration, and rest duration.

For paired data, the tuple includes both the word and the corresponding musical attributes.

This format ensures that the large language model can understand how lyrics and melody correspond to each other, enabling the generation of fully aligned song compositions. It also makes the processing of both pure lyrics and pure melody seamless.

5.2 Data Handling and Processing Units: Pretraining Dataset

Pretraining Dataset (SongCompose-PT)

The Pretraining Dataset module is crucial for building SongGen's foundational understanding of music and language. The dataset is divided into three parts:

Pure Lyrics Dataset: Contains over 283,000 songs in English. This helps the model learn how to generate coherent and contextually relevant lyrics.

Pure Melody Dataset: Consists of 20,000 melodies extracted from MIDI files. Using symbolic representations of pitch, duration, and rest, the dataset helps the model understand the structure of melodies without needing raw audio data.

Paired Lyric-Melody Dataset: Features 15,000 paired examples where lyrics and melodies are aligned at the word level. This is a critical part of the dataset,

as it teaches the model how to pair lyrics with a melody, enabling tasks like lyric-to-melody generation and vice versa.

To enhance the dataset's quality, techniques like data augmentation (e.g., pitch shifting) are used to introduce variety and improve the model's ability to generalize across different musical styles.

5.3 Training and Learning Procedures: Two-Stage Training Paradigm

Two-Stage Training Paradigm

The Two-Stage Training Paradigm consists of two main phases:

Pretraining Stage:

In this stage, the model learns fundamental concepts of music and language by processing vast amounts of pure lyrics, pure melodies, and paired data.

The goal of pretraining is to give the model a strong baseline understanding of how lyrics and melody function, both separately and together.

A key method here is next-token prediction, where the model learns to predict the next word or note based on previous inputs, thereby improving its ability to generate sequences.

Supervised Fine-Tuning Stage:

After pretraining, the model undergoes fine-tuning with specific tasks that mimic real-world use cases, such as generating lyrics for a given melody or completing an unfinished song.

The QA-style instruction-following tasks (10,000 pairs) provide the model with diverse challenges and ensure that it can follow flexible instructions, enabling it to compose coherent and contextually appropriate music.

This two-stage approach ensures that SongGen learns not only the basics of music and language but also how to generate, extend, and align lyrics and melody based on complex instructions.

5.4 Specialized Tools: Music Source Separation & Dynamic Time Warping

Music Source Separation

The Music Source Separation module focuses on extracting clean vocal tracks from audio data, separating the vocal melody from background accompaniment. This is done using tools like UVR (Ultimate Vocal Remover), which isolates the vocal part of the song. The separated vocals are then used for more accurate melody extraction, ensuring high-quality training data for the model.

This is particularly important when building paired datasets where clean melody data is needed to align with the corresponding lyrics. By removing the instrumental background, the model can focus solely on the relationship between the vocal melody and the lyrics.

Dynamic Time Warping (DTW) Algorithm

The Dynamic Time Warping (DTW) module is used to ensure accurate word-to-note alignment. This algorithm allows for the precise matching of lyrics with their corresponding musical notes by analyzing and aligning sequences based on their time stamps.

DTW calculates the optimal alignment path between the melody and the lyrics, even if the two sequences differ in timing or tempo.

This is essential for paired lyric-melody data, where each word must be matched to a note or group of notes in the melody.

By using DTW, SongGen can handle the variability in timing between lyrics and music, ensuring that generated songs are coherent and natural, with the lyrics fitting seamlessly with the melody.

CHAPTER 6

SYSTEM REQUIREMENT

CHAPTER 6

6. SYSTEM REQUIREMENT

The SongGen project is designed to efficiently handle the complex task of generating lyrics and melodies using advanced AI models. To ensure optimal performance, it requires a robust setup consisting of both software and hardware components tailored for high computational workloads.

6.1 Introduction

The system requirements for SongGen aim to support real-time lyric and melody generation using advanced AI techniques. The system is designed for high performance in handling symbolic music representations, ensuring efficiency in both training and real-time song composition tasks. The following are the necessary software and hardware components to run SongGen effectively.

6.2 Software Requirements

Operating System:

Linux-based operating system (Ubuntu 20.04 LTS or later) for server deployment.
Windows 10/11 or macOS (latest version) for development and testing.

Development Environment:

Python 3.8+: Required for running the model and related scripts.
Pytorch 1.12+: For training and deploying the SongGen model.
CUDA 11.3: For GPU-accelerated training and inference.

Libraries and Dependencies:

Transformers (Hugging Face): For utilizing pre-trained models and fine-tuning them.

Pretty MIDI: For handling and processing MIDI files.

Librosa: For additional music analysis and feature extraction.

Numpy, Scipy, and Pandas: For numerical and data handling operations.

Matplotlib and Seaborn: For visualizing model outputs and dataset statistics.

Generative AI Tools:

OpenAI API (for GPT integration) or any custom large language model based on GPT architecture for generating lyrics and melodies.

Backend Services:

FastAPI or Flask: For developing APIs to integrate the SongGen model with the user interface.

PostgreSQL: For managing song data, including lyrics, melodies, and paired datasets.

Docker: For containerizing the application to ensure a consistent deployment environment.

6.3 Hardware Requirements

Server Hardware (For Training and Production):

GPU: NVIDIA A100 (40 GB VRAM) or equivalent, with support for CUDA 11.x for large-scale model training.

CPU: Intel Xeon Processor with at least 12 cores and a 3.0 GHz clock speed for parallel data processing.

RAM: Minimum of 64 GB for handling large datasets and high concurrency during model training and inference.

Storage: 2 TB SSD for storing the SongCompose-PT dataset and model checkpoints. High-speed storage is essential for loading and processing large amounts of song data efficiently.

Client-Side Hardware (For Development and Testing):

CPU: Intel i7 or AMD Ryzen 7 processor with multi-threading support.

GPU: NVIDIA RTX 3080 (10 GB VRAM) or equivalent for faster model inference during development.

RAM: 16 GB minimum (32 GB recommended) for running tests and simulations without performance bottlenecks.

Storage: Minimum 500 GB SSD for local storage of datasets and pre-trained models.

CHAPTER 7

CODE

CHAPTER 7

CODE

```
import pretty_midi

import re

import numpy as np

import json

import torch

base_tones = {

    'C' : 0, 'C#': 1, 'D' : 2, 'D#': 3,

    'E' : 4, 'F' : 5, 'F#': 6, 'G' : 7,


    'G#': 8, 'A' : 9, 'A#':10, 'B' :11,

}

line_index = {

    0: 'first', 1 : 'second', 2: 'third',

    3 : 'fourth', 4 : 'fifth',

    5: 'sixth', 6 : 'seventh',

    7: 'eighth', 8 : 'ninth', 9: 'tenth',

}
```

```
def log_discretize(x, bins=512):
    eps = 1
    x_min = np.log(eps-0.3)
    x_max = np.log(6+eps)
    x = min(6, x)
    x = max(-0.3, x)
    x = np.log(x+eps)
    x = (x-x_min) / (x_max-x_min) * (bins-1)
    return np.round(x).astype(int)
def reverse_log_float(x, bins=512):
    if x == 79:
        return 0
    eps = 1
    x_min = np.log(eps-0.3)
    x_max = np.log(6+eps)
    x = x * (x_max - x_min)/(bins-1) + x_min
    x = np.exp(x) - eps
    return float("{:.3f}".format(x))
def bin_time(list_d):
    bin_list = []
```

```
    for item in list_d:
        if not isinstance(item, str):
            item = str(item)
        item_tuple = item.split(' ')
        out = ''
        for item_str in item_tuple:
            item_num = float(item_str)
            # out += f'<{item_num}>'
            bin = log_discretize(item_num)
            out += f'<{bin}>'
        bin_list.append(out)
    return bin_list
def append_song_token(model, tokenizer, config):
    old_token_len = len(tokenizer)
    new_tokens = ['<bol>','<bom>','<bop>','<eol>','<eom>','<eop>']
    for note in base_tones:
        for i in range(-1, 10): # -1 -> 9
            new_tokens.append(f'<{note}{i}>')
    for t_bin in range(512):
        new_tokens.append(f'<{t_bin}>')
```

```
    new_tokens = set(new_tokens) - set(tokenizer.get_vocab().keys())

    new_tokens = list(new_tokens)

    new_tokens.sort()

    tokenizer.add_tokens(new_tokens)

    new_token_len = len(tokenizer)

    model.tokenizer = tokenizer

    weight = nn.Parameter(torch.empty((new_token_len-old_token_len,
config.hidden_size)))

    nn.init.kaiming_uniform_(weight, a=math.sqrt(5))

    model.config.vocab_size = new_token_len

    model.output.weight.data = torch.cat([model.output.weight,
weight.to(model.device)], dim=0)

    model.output.weight.requires_grad = True

    new_token_embed = torch.randn(new_token_len-old_token_len,
config.hidden_size)

    new_weight = torch.cat([model.model.tok_embeddings.weight,
new_token_embed.to(model.device)], dim=0)

    model.model.vocab_size = new_token_len

    model.model.tok_embeddings.weight.data = new_weight

    model.model.tok_embeddings.weight.requires_grad = True

    return model, tokenizer
```

```
def tuple2dict(line):
    order_string = ['first', 'second', 'third', 'fourth', 'fifth', 'sixth', 'seventh', 'eighth', 'ninth', 'tenth']
    line = line.replace(" ", "")
    line = line.replace("\n", "")
    line = re.sub(r'\. |\.', '', line)
    # line = re.sub(r'The\d+line:', ' |', line)
    for string in order_string:
        line = line.replace(f'The{string}line:', ' |')
    special_pattern = r'<(.*?)>'
    song = {'lyrics':[], 'notes':[], 'notes_duration':[], 'rest_duration':[], 'pitch':[], 'notes_dict': [], 'rest_dict': []}
    for item in line.split('|')[1:]:
        x = item.split(',')
        notes = re.findall(special_pattern,x[1])
        note_ds = re.findall(special_pattern,x[2])
        rest_d = re.findall(special_pattern,x[3])[0]
        assert len(notes)== len(note_ds), f"notes:{'|'.join(notes)}, note_ds:{'|'.join(note_ds)}"
        for i in range(len(notes)):
            if i == 0:
```

```
            song['lyrics'].append(x[0])
        else:
            song['lyrics'].append('-')
        song['notes'].append(notes[i])
        song['pitch'].append(int(pretty_midi.note_name_to_number(notes[i])))
        song['notes_duration'].append(reverse_log_float(int(note_ds[i])))
        song['notes_dict'].append(int(note_ds[i]))
        if i == len(notes)-1:
            song['rest_duration'].append(reverse_log_float(int(rest_d)))
            song['rest_dict'].append(int(rest_d))
        else:
            song['rest_duration'].append(0)
            song['rest_dict'].append(0)
    return song
def dict2midi(song):
    # new_midi = pretty_midi.PrettyMIDI(charset="utf-8")#
    new_midi = pretty_midi.PrettyMIDI()
    instrument = pretty_midi.Instrument(program=0)
    # print(len(song["notes"]))
    current_time = 0  # Time since the beginning of the song, in seconds
```

```
    pitch = []
    for i in range(0, len(song["notes"])):
        #add notes
        notes_duration = song["notes_duration"][i]
        note_obj = pretty_midi.Note(velocity=100,
pitch=int(pretty_midi.note_name_to_number(song["notes"][i])), start=current_time,
                            end=current_time + notes_duration)
        instrument.notes.append(note_obj)
        #add lyrics
        # lyric_event = pretty_midi.Lyric(text=str(song["lyrics"][i])+ "\0",
time=current_time)
        # new_midi.lyrics.append(lyric_event)
        current_time +=  notes_duration + song["rest_duration"][i]# Update of the time
    new_midi.instruments.append(instrument)
    lyrics = ' '.join(song["lyrics"])
    return new_midi, lyrics
def gen_midi(line, file_name):
    song  = tuple2dict(line)
    #song['lyrics'] = ['I','-','you','-','I','-','you','-','I','-','you','-','he','-']
    new_midi, lyrics = dict2midi(song)
    # save midi file and lyric text
```

```
    new_midi.write(file_name+'.mid')
    with open(file_name+'.txt', "w") as file:
        file.write(lyrics)
    print(f'midi saved at ~/{file_name}.mid, lyrics saved at ~/{file_name}.txt')
from transformers import AutoTokenizer, AutoModel
ckpt_path = "Mar2Ding/songgen_sft"
tokenizer = AutoTokenizer.from_pretrained(ckpt_path, trust_remote_code=True)
model = AutoModel.from_pretrained(ckpt_path,
trust_remote_code=True).cuda().half()
prompt = ‘Write an energetic English song about a powerful movement that shakes the world and inspires everyone to join in.’
model.inference(prompt, tokenizer)
```

CHAPTER 8

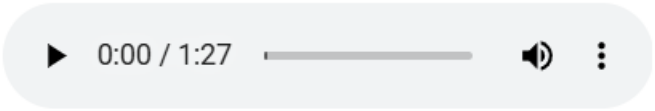
OUTPUT

CHAPTER 8

OUTPUT

```
<verse>
we came in like a northern storm blew y'all doors off
hung the world upside down
and now y'all better know the drill
if you're not with us then you must be against us
we won't stop until
we're worldwide until everybody singing this our theme our theme song
we gotta keep the fire burning even though
it's harder to keep the flame alive inside

<chorus>
when you're calling my name out
when you're calling my name out
when you're calling my name out
when you're calling my name out
we came in like a northern storm blew y'all doors off
hung the world upside down
hung the world upside down
```



CHAPTER 9

WORK AND IMPLEMENTATION

CHAPTER 9

9. WORK AND IMPLEMENTATION

This chapter provides a detailed description of the working and implementation of SongGen, focusing on the methodologies and processes involved in developing and training the model. SongGen is designed to leverage large language models (LLMs) for the purpose of composing songs, integrating both lyrics and melodies in a coherent manner. The implementation is divided into two main aspects: symbolic representation and the two-stage training paradigm.

9.1 Symbolic Representation for LLM

9.1.1 Tuple Data Format

To efficiently feed lyrics and melodies into the LLM, we developed a novel tuple-based format. This format addresses the challenges of aligning lyrics and melodies, which traditional sequential approaches struggle with. The tuple format organizes the data into discrete musical units, each representing either a lyric, a melody, or a lyric-melody pair.

Pure Lyric Data: Each tuple contains a single word.

Pure Melody Data: Each tuple includes the note pitch, note duration, and rest duration.

Lyric-Melody Pairs: Tuples merge the lyric word and corresponding note elements, where a single word may correspond to multiple notes.

We use a vertical bar (|) to separate different tuples, facilitating the model’s understanding of the correspondence between lyrics and melody.

9.1.2 Discretized Duration

To effectively process duration information, we employ a logarithmic encoding scheme to categorize note durations into predefined bins. This approach transforms continuous duration ranges into a set of discrete values, making the input more organized and concise for the LLM. Specifically, we set the duration range to $[-0.3,6]$ seconds, divided into 512 bins. The encoding and decoding processes ensure that the model can handle timing variations accurately.

9.1.3 Vocabulary Extension

To represent the discretized time units and note values within the model's vocabulary, we introduce auxiliary tokens, denoted by $\langle \cdot \rangle$ symbols. This extension includes 512 unique tokens for temporal discretization and 120 distinct musical tokens for pitch classes across 10 octaves. This comprehensive vocabulary enables the model to understand and generate detailed musical content.

9.2 Two-Stage Training Paradigm

The training of SongGen involves two main stages: pretraining and supervised fine-tuning. This two-stage approach ensures that the model learns both the fundamentals of music and language, as well as specific song generation tasks.

9.2.1 Pretraining Stage

The pretraining stage focuses on enriching the model with fundamental musical knowledge. We use the SongCompose-PT dataset, which includes pure lyrics, pure melodies, and paired lyric-melody data.

Data Tokenization: Cleaned and processed lyrics and melody data are converted into tuples.

Next Token Prediction: The model learns the basic structure and patterns in both lyrics and melodies through this task. The dataset is augmented with techniques like pitch shifting to enhance learning.

To balance the learning process, we use an equal amount of pure melody, pure lyric, and paired data, maintaining a 1:1:1 sample ratio. The total dataset consists of 0.23 billion tokens for lyrics, 0.28 billion tokens for melodies, and 0.16 billion tokens for paired data.

9.2.2 Supervised Fine-Tuning Stage

After pretraining, the model is fine-tuned using instruction-following QA pairs to handle specific song-generation tasks, including:

Lyric-to-Melody Generation

Melody-to-Lyric Generation

Song Continuation

Text-to-Song Creation

For these tasks, we manually prepare 3,000 QA pairs for the first three tasks and use GPT-4 to produce 1,000 song summaries for text-to-song generation. This curated data enhances the model's ability to generate coherent and contextually appropriate songs.

CHAPTER 10
CONCLUSION AND FUTUREWORK

CHAPTER 10

10. CONCLUSION AND FUTUREWORK

SongGen leverages large language models to create cohesive songs by integrating symbolic song representations and a robust training process. It outperforms advanced LLMs like GPT-4 in various song generation tasks. Enhancements include expanding training data, improving real-time generation capabilities, developing interactive interfaces, refining evaluation metrics, and ensuring fairness and accessibility.

10.1 Conclusion

The development of SongGen marks a significant advancement in the field of automated music composition, leveraging the capabilities of large language models (LLMs) to generate coherent and harmonious songs. By integrating symbolic song representations, SongGen effectively bridges the gap between lyrical and melodic creation, offering a versatile tool for musicians and composers.

Key achievements of SongGen include:

Innovative Data Representation: The use of tuple-based formats and discretized duration encoding ensures precise alignment between lyrics and melodies.

Comprehensive Training Paradigm: A robust two-stage training process, involving extensive pretraining on a large-scale dataset (SongCompose-PT) and fine-tuning with instruction-following QA pairs, equips the model to handle diverse song generation tasks.

Superior Performance: Through rigorous objective and subjective evaluations, SongGen demonstrated superior performance in tasks such as lyric-to-melody

generation, melody-to-lyric generation, song continuation, and text-to-song creation, outperforming advanced LLMs like GPT-4.

The success of SongGen highlights the potential of integrating advanced NLP techniques with music generation, paving the way for innovative applications in both the music industry and creative arts.

10.2 Future Works

Despite its notable achievements, there are several avenues for future enhancement of SongGen:

Expansion of Training Data:

Diverse Genres and Languages: Incorporating more diverse musical genres and languages into the training dataset can improve the model's versatility and adaptability to different musical styles.

Higher-Quality Paired Datasets: Increasing the size and quality of paired lyric-melody datasets will further enhance the model's ability to generate well-aligned songs.

Model Enhancements:

Advanced Pretraining Techniques: Exploring more sophisticated pretraining techniques, such as multi-task learning and transfer learning, can provide the model with a deeper understanding of musical structures.

Real-Time Generation Capabilities: Improving the model's efficiency and reducing latency to enable real-time song composition and interactive applications.

User Interface and Interaction:

Interactive Composition Tools: Developing user-friendly interfaces that allow users to interact with the model, providing real-time feedback and customization options for song generation.

Integration with Music Production Software: Seamlessly integrating SongGen with popular music production software to facilitate easy adoption by musicians and producers.

Enhanced Evaluation Metrics:

Comprehensive Evaluation Frameworks: Developing more sophisticated evaluation frameworks that consider a wider range of musical attributes and user feedback to assess the quality of generated compositions more holistically.

Ethical Considerations and Accessibility:

Fairness and Bias Mitigation: Ensuring that the model's training data and generation processes do not propagate biases, promoting fairness and inclusivity in generated content.

Accessibility Features: Incorporating features that make the tool accessible to users with disabilities, ensuring that the benefits of SongGen are widely available.

By addressing these areas, future iterations of SongGen can become even more powerful and versatile, driving further innovation in the intersection of artificial intelligence and music composition. The ongoing evolution of this technology promises to transform the creative processes in music, offering new possibilities for artists and enthusiasts alike.

CHAPTER 11
REFERENCES

CHAPTER 11

11. REFERENCES

[1] Agostinelli, A., Denk, T. I., Borsos, Z., Engel, J., Verzetti,
M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., et al. Musiclm: Generating music from text. arXiv preprint arXiv:2301.11325, 2023.

[2] Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., et al. Qwen technical report. arXiv preprint arXiv:2309.16609, 2023.

[3] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra,
G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. Journal of Machine Learning Research, 24(240):1–113, 2023.

[4] Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and D´ efossez, A. Simple and controllable music generation. In Thirty-seventh Conference on Neural Information Processing Systems, 2023.

[5] Huang, Q., Park, D. S., Wang, T., Denk, T. I., Ly, A., Chen, N., Zhang, Z., Zhang, Z., Yu, J., Frank, C., et al. Noise2music: Text-conditioned music generation with diffusion models. arXiv preprint arXiv:2302.03917, 2023.

[6] Hussain, A. S., Liu, S., Sun, C., and Shan, Y. M2ugen: Multi-modal music understanding and generation with the power of large language models. arXiv preprint arXiv:2311.11255, 2023.

[7] Kondratyuk, D., Yu, L., Gu, X., Lezama, J., Huang, J., Hornung, R., Adam, H., Akbari, H., Alon, Y., Birodkar, V., et al. Videopoet: A large language model for zero shot video generation. arXiv preprint arXiv:2312.14125, 2023.

[8] Schneider, F., Jin, Z., and Sch¨olkopf, B. Mo\ˆ usai: Text to-music generation with long-context latent diffusion. arXiv preprint arXiv:2301.11757, 2023.

[9] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and finetuned chat models. arXiv preprint arXiv:2307.09288,
2023.

[10] Yang, A., Xiao, B., Wang, B., Zhang, B., Bian, C., Yin, C., Lv, C., Pan, D., Wang, D., Yan, D., et al. Baichuan2: Open large-scale language models. arXiv preprint arXiv:2309.10305, 2023.